Knowledge web 2.2: Heterogeneity in the semantic web

Coordinator: Jérôme Euzenat (INRIA)
Thanh Le Bach, Rose Dieng, Chan Le Duc, Amedeo Napoli, Antoine Zimmermann (INRIA), Rubn Lara, Adrian Mocan, François Scharffe (U. Innsbruck), Paolo Bouquet, Luciano Serafini, Pavel Shvaiko, Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, Ilya Zaihrayeu (University of Trento), Marc Ehrig, York Sure, Pascal Hitzler, Markus Krötzsch (U. Karlsruhe), Anja Jentzsch, Malgorzata Mochol (Free University of Berlin), Heiner Stuckenschmidt (University of Mannheim), Andreas Hess, Willem Robert van Hage (Vrije Universiteit Amsterdam), Ondřej Šváb, Vojtěch Svátek (University of Economics, Prague) Wei Hu, Ningsheng Jian, Gong Cheng and Yuzhong Qu (Southeast University China), George Stoilos, George Stamou (ITI-Certh), Umberto Straccia (ISTI-CNR), Raphaël Troncy (CWI Amsterdam), Petko Valtchev (Université de Montréal), Jess Barrasa, Ral Garca Castro (UP Madrid), Enrico Franconi, Sergio Tessaris (FU Bolzano) Jan De Bo, Mustafa Jarrar, Sven Van Acker (VU Brussels), Manfred Hauswirth (EPF Lausanne), Diana Maynard (Sheffield U.)

Abstract.

Keyword list: ontology alignment, alignment life cycle, alignment edition, ontology merging, ontoloy transformation, data translation, query mediation, reasoning, alignment support, schema matching, schema mapping, alignment language, expressive alignments, alignment semantics, ontology matching, ontology alignment, , data integration, peer-to-peer, do2002a, benchmarking, contest, performance measure, ontology mapping, mediation, format, interoperability, OWL, SWRL, SBO, Alignment API, C-OWL, SEKT-ML, SKOS.

Document Identifier	KWEB/2004/WP2.2/synthesis/0.1
Project	KWEB EU-IST-2004-507482
Version	0.1
Date	December 19, 2007
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas
Ecole Polytechnique Fédérale de Lausanne
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
National University of Ireland Galway
Universidad Politècnica de Madrid
University of Innsbruck
University of Karlsruhe
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel

Changes

Version	Date	Author	Changes
0.1	13.11.2007	Jérôme Euzenat	initial compilation

Forework

This document is a compilation of deliverable of the "heterogeneity" (2.2) work package of the *K*nowledge web network of excellence (2004-2007). The raw material from deliverable has been rearranged in a coherent outline.

Not all the deliverable content are compiled here because some of them were building on top of another. In each case, we choose the more advanced material. Usually, content such as the executive summary, introduction and conclusion has been omitted.

In addition, this documents lacks the figures drawn in XYpic for Chapter 18. They have been replaced by the mention: CURRENTLY UNAVAILABLE.

Executive Summary

2.2.1 Summary Definition of a common framework for characterizing alignment of heterogeneous information. This report describes various approaches towards this goal and shows the relations between them. It also provides a description of the alignment sructure and process.

2.2.3 Summary In this document we provide an overall view of the state of the art in ontology alignment. It is organised as a description of the need for ontology alignment, a presentation of the techniques currently in use for ontology alignment and a presentation of existing systems. The state of the art is not restricted to any discipline and consider as some form of ontology alignment the work made on schema matching within the database area for instance.

2.2.2 Summary This document considers potential strategies for evaluating ontology alignment algorithms. It identifies various goals for such an do2002a. In the context of the Knowledge web network of excellence, the most important objective is the improvement of existing methods. We examine general do2002a strategies as well as efforts that have already been undergone in the specific field of ontology alignment. We then put forward some methodological and practical guidelines for running such an do2002a.

2.2.4 and 2.2.9 Summary This deliverable presents the do2002a campaign carried out in 2005 and the improvement participants to these campaign and others have to their systems. We draw lessons from this work and proposes improvements for future campaigns. This document discusses its latest advances in ontology matching do2002a: (i) by providing new measures for ontology matching do2002a, such as semantic precision and recall and (ii) by presenting the results of the Ontology Alignment Evaluation Initiative: OAEI-2006 campaign.

1.2.2.2.1 Summary Choosing a matching tool adapted to a particular application can be very difficult. This document analyses the choice criteria from the application viewpoint and their fulfilment by the candidate matching systems. Different methods (paper analysis, questionnaire, empirical do2002a and decision making techniques) are used for assessing them. We evaluate how these criteria can be combined and how they can help particular users to decide in favour or against some matching system.

2.2.6 and 2.2.10 Summary This deliverable focusses on the definition of a delivery alignment format for tools producing alignments (mapping tools). It considers the many formats that are currently available for expressing alignments and evaluate them with regard to criteria that such formats would satisfy. It then proposes some improvements in order to produce a format satisfying more needs. This deliverable provides the description of an alignment language which is both expressive and independent from ontology languages. It defines the language through its abstract syntax and semantics depending on ontology language semantics. It then describes two concrete syntax: an exchange syntax in RDF/XML and a surface syntax for human consumption. Finally, it presents the current implementation of this expressive language within the Alignment API taking advantage of the OMWG implementation.

2.2.7 Summary Dealing with heterogeneity requires finding correspondences between entities of ontologies and using these correspondences for performing some action such as merging ontologies, transforming ontologies, translating data, mediating queries and reasoning with the aligned ontologies. This deliverable considers this problem through the introduction of an alignment life cycle which also identifies the need for manipulating, storing and sharing the alignments before processing them. In particular, we also consider support for run-time and design-time alignment processing.

In Knowledge web, we have considered the heterogeneity problem as a two step problem: (i) matching ontologies to determine alignments and (ii) processing alignments according to application needs. So far most of the work has been dedicated to the ontology matching problem.

In this deliverable, we introduce an alignment life cycle, tied to the ontology life cycle. This life cycle identifies five different operations applied to alignments: creation, do2002a, enhancement, storing and sharing, and finally processing. Since we covered elsewhere the two first operations, this deliverable focusses on the three last ones.

In particular, this mandates supporting applications to deal with alignments at design-time and at run-time. Indeed, the fact that some applications require high quality alignments (or high quality programs to apply) makes that it is impossible to compute these at runtime. They must be either stored somewhere and retrieved dynamically or computed definitively at design-time. On the other hand, some applications evolve in a so dynamic world that this is impossible and they have to compute alignments dynamically.

We also review the kind of operations that can be carried out for dealing with heterogeneity. This includes merging ontologies, transforming ontologies, translating data, mediating queries and reasoning with aligned ontologies. There usually exist a few such systems in each category, most of them tied to a particular matching strategy and thus closely related to particular applications. These systems cannot take advantage of more general alignment support.

In contrast, tools developed with the two step strategy can take advantage of the results of the many matching systems available and provide executable input for any of the processing operations. For each kind of operation, we show how they can be developed from alignments. We mention some of these tools and, in particular, those developed by Knowledge web partners such as the Alignment API and WSMT.

Contents

Part I

Problem definition

Chapter 1

Introduction

1.1 Objectives of this document

The goal of this document is to provide a common framework for future work in the domain of semantic interoperability. In this document the problem of overcoming heterogeneity is equated to the problem of discovering, expressing and using mappings across ontologies.

The main contribution is a general characterization of what an alignment and an alignment process are in the context of the Semantic Web enterprise, and a specification of a formal semantics. The framework aims at the greatest possible generality, but will not cover approaches and models which fail to fulfill a high level requirement of any semantic web development, namely that mappings – the result of the alignment process – should have an explicit and formal semantics, as this is the minimal conditions for their usability in any semantic-based application.

This common framework is to be used by the partners of the Knowledge Web work package 2.2 and other Knowledge web groups as a reference document for models, languages and techniques related to the problem of aligning heterogeneous information. It is not a goal of this document to provide any detail or model of how a mapping between heterogeneous representations (e.g. ontologies) can be discovered, nor how mappings can be used in any specific application (e.g., data integration, query answering). The actual discovery and usage of mappings is the object of deliverable D2.2.3. Deliverable 2.2.2 will use this framework for characterising benchmark tests.

1.2 Terminology

The framework presented in this document builds on top of a lot of recent work on the problem of semantic interoperability. In this area, different authors use different words to refer to similar concepts, and vice versa sometimes different concepts are referred to by the same name. In this section, we provide a tentative and partial glossary with the definition of terms as they will be used in the rest of the document and should be used within the Knowledge web work package 2.2.

Mapping: a formal expression that states the semantic relation between two entities belonging to different ontologies. When this relation is oriented, this corresponds to a restriction of the usual mathematical meaning of mapping: a function (whose domain is a singleton). Mappings are discussed at length in Chapter 16.

Ontology Alignment: a set of correspondences between two or more (in case of multi-alignment) ontologies (by analogy with DNA sequence alignment). These correspondences are expressed as mappings. Alignments are detailed in Chapter 3.

Ontology Coordination: broadest term that applies whenever knowledge from two or more ontologies must be used at the same time in a meaningful way (e.g. to achieve a single goal).

Ontology Transformation: a general term for referring to any process which leads to a new ontology o' from an ontology o by using a transformation function t. Transformations and the like are the subject of further work in this work package.

Ontology Translation: an ontology transformation function t for translating an ontology o written in some language L into another ontology o' written in a distinct language L'.

Ontology Merging: the creation of a new ontology o_m from two (possibly overlapping) source ontologies o' and o''. This concept is closely related to that of *integration* in the database community.

Ontology Reconciliation: a process that harmonizes the content of two (or more) ontologies, typically requiring changes on one of the two sides or even on both sides [Hameed *et al.*, 2004].

Meaning Negotiation: the protocol through which two agents (either human or artificial) agree on the changes required to reconciliate their ontologies.

1.3 Structure of the document

This deliverable will first consider the types of heterogeneity that may occur in the semantic web and how to overcome them through alignment (Chapter 2). It will then propose a general structure (Chapter 3) for these alignments which is exploited in a categorical framework (Chapter 18). The semantics is then refined in the context of a model theoretic characterisation of mappings in distributed systems (Chapter 16). The framework ends with a characterization of the alignment process (Chapter 5).

Chapter 2

Semantic heterogeneity

In a distributed and open system (like the semantic web), heterogeneity cannot be avoided. Different actors have different interests and habits, use different tools, and use knowledge at different levels of detail. These various reasons for heterogeneity lead to different forms of heterogeneity that are considered below.

An ontology is a set of assertions that are meant to model some particular domain, in a consensual way. However, there is a huge body of evidence (in the literature of artificial intelligence, Cognitive Science, Linguistics, Epistemology, Sociology of Knowledge) that an ontology – as any other explicit representation of knowledge – always depends on a collection of implicit assumptions, no matter how hard its designers work to make it as "objective" as possible. These assumptions (including its designer's goals, background knowledge, biases, etc.) have the effect of creating several forms of heterogeneity between ontologies, even between ontologies on the same domain. We classify below (Section 2.1) the main forms of heterogeneity.

Heterogeneity affects the ontology used as well as the data exchanged. However, the actors of the semantic web have to communicate and to collaborate and thus need to mitigate this kind of heterogeneity. We consider then how alignments can be used for overcoming these problems (Section 2.2).

2.1 Forms of heterogeneity

Heterogeneity may occur at different levels, and a detailed list of all forms of possible mismatches is beyond the scope of this document (see [Giunchiglia and Walsh, 1992; Benerecetti *et al.*, 2000; Klein, 2001; Euzenat, 2001; Corcho, 2004; Hameed *et al.*, 2004; Ghidini and Giunchiglia, 2004]). However, for the sake of the definition of a common framework, we suggest that they can be classified into four main levels: syntactic, terminological, conceptual, semiotic/pragmatic. Each of them is briefly described in the following sections.

2.1.1 The syntactic level

At the syntactic level, we encounter all forms of heterogeneity that depend on the choice of the representation format. Indeed, there are several proposed formats for ontology representation (e.g. OWL, KIF), and each of them is based on a different syntax.

Some of them are syntactic sugar (e.g., n3 and RDFS, DATALOG and a subset of Prolog), some of them are more complicated and involve expressing the same thing (having the same set of models) through totally different syntax.

Example 1 (Translating from DLR to CPDL)**.** *In order to decide query containment in the DLR description logics, [Calvanese* et al.*, 1998a] defines a mapping from the DLR logic (which introduces n-ary relations) to the CPDL logic (Propositional Dynamic Logic with Converse). These relations are represented by concepts with exactly n features to the components of the relation.*

This transformation is a consequence preserving syntactic transformation.

In this document, and more generally in the work package 2.2 of Knowledge web, we are not strongly concerned about this syntactic level, which is well understood in computer science in general. Therefore, in what follows, we will assume that the different formats can be interoperated at a syntactic level. This is typically achieved through a translation function (see Section 1.2).

As a working assumption, from now on we assume that ontologies are represented using a common syntax (e.g., OWL). However, the framework presented in this document does not depend in any essential way on this assumption.

2.1.2 The terminological level

At the terminological level, we encounter all forms of mismatches that are related to the process of naming the entities (e.g. individuals, classes, properties, relations) that occur in an ontology. Naming is the process of associating a linguistic object from a public language (namely a language that is then use to exchange information with other parties) to entities described in an ontology. This level should not be confused with the conceptual level (see below); indeed, tricky terminological mismatches may occur in situations where the involved ontologies are conceptually equivalent.

Typical examples of mismatches at the terminological level are:

- different words are used to name the same entity (*synonymy*);
- the same word is used to name different entities (*polysemy*);
- words from different languages (English, French, Italian, Spanish, German, Greek, etc.) are used to name entities;
- syntactic variations of the same word (different acceptable spellings, abbreviations, use of optional prefixes or suffixes, etc.).

In a sense, mismatches at the terminological level are not as deep as those occurring at the conceptual level (see below). However, we should notice that most real cases have to do with the terminological level (e.g., with the way different people name the same entities), and therefore this level is at least as crucial as the other one.

2.1.3 The conceptual level

At the conceptual level, we encounter mismatches which have to do with the content of an ontology. Discrepancies at this level can be analyzed in two main classes:

- *metaphysical* differences, which have to do with how the world is “broken into pieces” (i.e., what entities, properties and relations are represented in an ontology);
- *epistemic* differences, which have to do with the assertions that are made about the selected entities.

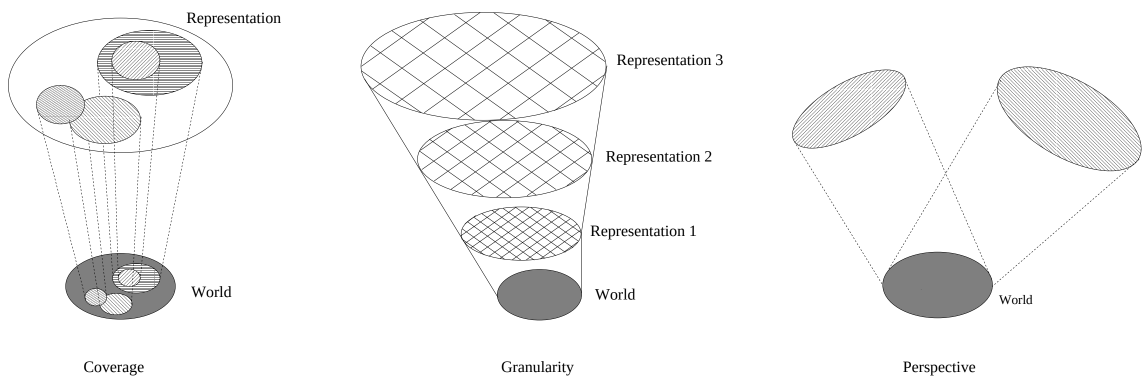


Figure 2.1: The three dimensions of heterogeneity at the conceptual level

These two kinds of differences explain, for example, why different ontologies of the same domain may start from different primitive classes, or why different ontologies may contain different (possibly contradictory) assertions about the same entities. As epistemic differences cannot be dealt with exclusively through mappings, in the rest of the document we will ignore this kind of difference, but we'll comment a bit later on the relation between the two forms of heterogeneity.

The practical forms in which metaphysical differences can arise are countless. However, following the artificial intelligence literature in this topic (in particular [Benerecetti *et al.*, 2000]), we suggest to cluster them into three abstract types:

Coverage: an ontology may differ from another as they cover different portions – possibly overlapping – of the world (or even of a single domain). For example, an ontology on sport may include car racing, whereas another may decide to ignore it as part of the sport domain; an ontology may contains properties of car racing that another disregards; and so on.

Granularity: an ontology may differ from another as the first provides a more (or less) detailed description of the same entities. For example, an ontology concerned with accounting and taxes, or delivery, would only consider the generic concept of document, while an ontology for libraries or scholars would distinguish between types of documents, e.g. books, biographies or autobiographies. Likewise, in the ontology of a Finnish, or a nivologist, there are many concepts of snow depending on how it is, while the ontology of a Tahitian or computer scientist would include significantly fewer snow related concepts.

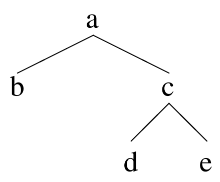
Perspective: an ontology may provide a viewpoint on some domain which is different from the viewpoint adopted in another ontology. For example, two ontologies may represent the same domain at the same level of coverage and granularity, but at different points in time (which means that the same property can hold at the time when the first ontology was designed and do not hold at the time when the other was designed, without a real epistemic disagreement), or from a different spatial perspective (what is on the right hand side from one agent's perspective may be to the left hand side for another agent facing the opposite direction).

Figure 2.1 provides a graphical representations of these three dimensions along which ontology may differ at the conceptual level.

2.1.4 The semiotic/pragmatic level

Finally, at the semiotic/pragmatic level, we encounter all the discrepancies that have to do with the fact that different individuals/communities may interpret the same ontology in different ways in different contexts.

For instance, in a context related to knowledge formalisation, a user can express knowledge under the form of class hierarchies and first order clauses and then communicate it by using an interoperability language. But if this last language expresses all the knowledge with clauses (though preserving the semantics of the assertions), the initial user will hardly recognise (and hardly understand) the semantically equivalent result (see figure 2.2). Hence, when a transformation translates between formal languages, good understanding cannot be ensured by meaning preservation (which can indeed be preserved in this case) [Euzenat, 2000; Bechhofer *et al.*, 2001]. In this case, there is no syntactic heterogeneity (because clauses are allowed in the initial model) and no conceptual nor terminological heterogeneity: only a failure to interpret the (equivalent) representation by its designer.



$$\begin{aligned}\forall x, b(x) &\implies a(x)\\ \forall x, c(x) &\implies a(x)\\ \forall x, d(x) &\implies c(x)\\ \forall x, e(x) &\implies c(x)\end{aligned}$$

Figure 2.2: Do these representations mean the same?

The intended usage has a great impact on alignment, as it can be quite risky to map entities onto each other only because they are semantically related. For example, if the concept `Europe` appears in the classification schema of a multimedia repository along a path such as `Images/B&W/Europe`, we should not conclude that it is equivalent to the concept of Europe in a geographic ontology, as the pragmatically determined meaning of the first is to be a container of black and white images of Europe, whereas the intended meaning of the second is the continent itself (this is not to say that the two things are not connected, but to warn that mappings should take intended use of each structure into account).

2.2 Overcoming heterogeneity

One common approach to the problems of heterogeneity is the definition of relations across the heterogeneous representations, in particular across ontologies. These correspondence can be used for various tasks such as merging ontologies, generating mediators, translating messages, etc. These relations can be used for transforming expression of one ontology into a form compatible with that of the other. This may happen at any level:

syntactic: through semantic-preserving transducers;
terminological: through functions mapping lexical information;
conceptual: through general transformation of the representations (sometimes requiring a complete prover for some languages);

pragmatic: through transformation taking care of the context. This path is hardly explored though ([Bouquet *et al.*, 2003b; Goguen, 1999] are exceptions).

This means that any mapping across ontologies has four distinct components:

1. a *syntactic component*: the syntactic transformations needed to transform one representation format into another. As we said, this aspect is not central in this document, and therefore we will ignore it;

2. a *terminological component*: the part of a mapping that expresses terminological relations between the expressions used to name the entities to be mapped. Simple examples are: the name of two entities is the same; the two entities are named with expressions that are one the translation of the other in a different language; the name of an entity is an abbreviation of the name of the other;

3. a *conceptual component*: the part of a mapping that expresses the relation between entities in different ontologies. Simple examples are: concept c_1 in ontology O_1 is equivalent to concept c_2 in ontology O_2; concept c_1 in ontology O_1 is similar to concept c_2 in ontology O_2; individual i_1 in ontology O_1 is the same as individual i_2 in ontology O_2…

4. a *semiotic/pragmatic component*: the part of a mapping that bridges the use of entities in different ontologies. For example, that the concept c_1 used in a schema O_1 to classify a collection of documents is used in a sense defined by c_2 in ontology O_2; that the name used for a concept in a schema is taken from a lexicon L.

In work package 2.2, we restrict our attention to terminological and conceptual heterogeneity. Indeed, syntactic heterogeneity is well understood in computer science and is generally solved by proving the semantic-preserving correspondence between two languages; pragmatic heterogeneity is currently a relatively poorly structured research domain (in which we contribute anyway). The techniques for finding, expressing and using alignments at the terminological and conceptual level are relatively integrated. Finally, the conceptual part is the most studied component of a mapping, and is probably the most important one for its role in the development of the semantic web.

The use of mappings across heterogeneous ontologies requires a clear definition of their meaning. Similarly, for generating useful alignments, it is better to know what is expected from them. This is the reason why the current framework will provide a syntax and semantics for the mappings which make correspondences.

Because we are concerned here with finding alignments, this framework provides two main elements:

– a general characterization of ontology alignments (§ 3) and a formal semantic for mappings (§ 16) that is to be used when using them as well as when finding them;
– a general definition of the alignment process (§ 5) and its potential dimensions (in particular more specific constraints applying to the resulting mapping).

These issues are covered in the next chapters.

Chapter 3

Ontology alignment and mapping

For our purposes, aligning two (or more) ontologies is a process that produces a set of mappings across ontologies which allow ontology coordination (see Glossary); said differently, mappings are tools that may enable the "flow" of information across heterogeneous ontologies by describing the relations existing between entities of different ontologies.

Following the analysis we proposed in Section 2.1, here we present a very abstract characterization of mappings (Section 3.1), and then propose a structure for these mappings (Section 3.2).

3.1 General characterization of alignments

Let o and o' be two ontologies. Then, following the analysis we provided in Section 2.1, the three basic relations between ontologies can be characterized as follows:

Coverage: the two ontologies describe different (possibly overlapping) regions of the world at the same level of detail and from a unique perspective (see left hand side of Figure 2.1);

Granularity: the two ontologies describe the same region of the world from the same perspective but at different levels of detail (see central part of Figure 2.1);

Perspective: the two ontologies describe the same region of the world, at the same level of detail, but from a different perspective (see right hand side of Figure 2.1).

With respect to this description, an alignment can be viewed as an operator $\alpha(o, o')$ that:

– given two ontologies o and o' with different coverage, tells us how the two ontologies can be used together to achieve a (less partial) description of the world;

– given two ontologies o and o' with different granularity, tells us how facts in o can be systematically translated into facts of o' (for example, how a fact f belonging to o can be rewritten as a logically equivalent fact f' in o');

– given two ontologies o and o' with different perspective, tells us how a fact f in o would be seen from the perspective of o'.

Let us briefly discuss the three categories and consider a few simple examples.

3.1.1 Aligning ontologies with different coverage

Two ontologies which differ only for the portion of the world they describe can be disjoint or may overlap. In the first case, as we excluded epistemic discrepancies (in particular, inconsistencies), they can easily be viewed as a partitioned theory, which can be jointly used to provide knowledge about the world. A simple example is an ontology o about soccer and an ontology o' about cricket. If we assume that there is no intersection between the two, an alignment will tell us that the two ontologies have no relation at all, and thus operations like inclusion, merge, and so on can be performed with no harm.

The situation is slightly different when the two theories (partially or completely) overlap. A simple example is an ontology o describing team sports and another describing indoor sports, where some sports (like volleyball) may belong to both ontologies. In this case, we must be able to recognize the common part and solve possible syntactic and terminological problems. Indeed, if we exclude inconsistencies, the only potential heterogeneity between the two ontologies may concern the syntactic format (e.g., RDF Schemas and OWL) and the choice of names used to identify the common entities (e.g., individuals, classes, and so on).

3.1.2 Aligning ontologies with different granularity

Let us now consider the case of two ontologies o and o' that describe the same portion of the world, but at different level of granularity. Simple examples are: when o characterizes the position of physical objects only by two coordinates (latitude and longitude), whereas o' takes into account also a third coordinate (height above the sea level); when o expresses measures in centimeters and o' in millimiters; and so on.

For granularity, the alignment should provide a way to move from the level of representation of an ontology to the level of representation of another ontology. Model-theoretically, this operation is more complex than the operation required in the previous case (coverage), as it requires to put in relation models which are intrinsically heterogeneous (e.g., facts of the form $loc(x, y)$ in o with facts of the form $loc(x, y, t)$ in o', where x and y may be expressed in different units of measure).

3.1.3 Aligning ontologies with different perspective

Finally, let us imagine that two ontologies o and o' describe the same region of the world, at the same level of granularity but from a different perspective. A very intuitive example is a representation using indexical expressions (like “here”, “I”, “now”, “yesterday”), as the content of such an expression essentially depends on where, when, from whom it is uttered, and their correct interpretation often requires the ability of *shifting* one’s perspective. But of course there are less direct examples. For example, the supporters of two different political parties will apply opposed descriptions to the same politicians; “cold” will be applied to different climatic conditions in Finland and in Greece; and so on.

In this case, alignment should provide a way of “rotating” the perspective of an ontology, or – as we said above – to shift its viewpoint. For some forms of heterogeneity, this can be done systematically and in a relatively simple way (e.g. for indexical descriptions); however, in general the change of perspective is a very hard task for any ontology alignment method.

3.2 Structure of a mapping

In this document, we propose to see alignment as a process that starts from two representations o and o' and produces a set of mappings between pairs of (simple or complex) entities $\langle e, e' \rangle$ belonging to O and O' respectively.

Intuitively, we will assume that in general a mapping can be described as a quadruple:

$$\langle e, e', n, R \rangle$$

where:

1. e and e' are the entities between which a relation is asserted by the mapping (e.g., formulas, terms, classes, individuals);
2. n is a degree of trust (confidence) in that mapping (notice, this degree does not refer to the relation R, it is rather a measure of the trust in the fact that the mapping is appropriate ("I trust 70% the fact that the mapping is correct/reliable/…"). The trust degree can be computed in many ways, including users' feedback or log analysis;
3. R is the relation associated to a mapping, where R identifies the relation holding between e and e'. Nothing is said about the relation but that it must apply to the pair of entities. For instance, this relation can be a simple set-theoretic relation (applied to entities seen as sets or their interpretation seen as sets), a fuzzy relation (see Section 16.4), a probabilistic distribution over a complete set of relations, a similarity measure, etc.

The degree of confidence must satisfy some minimum requirements, due to the model-theoretic semantics of the basic representation language like OWL. First of all, we require that the degrees are part of a structure $\langle D, \leq, \perp, \top \rangle$ such that D is the set of degrees, $\leq$ is an order on $D \times D$ such that whatever $d \in D$, $\perp \leq d \leq \top$. This structure is applicable to a wide number of measures (e.g., boolean lattice, fuzzy degrees, probabilities, any lattice). Moreover, whatever method is associated to the computation of the degree of confidence associated to some mapping, whenever such degree is $\perp$, then this must be interpreted as if the relation is logically false, and whenever such degree is $\top$, then this must be interpreted as if the relation be logically true. Last, we require some sort of smoothness, continuity and monotonicity of the degree of confidence function from zero to one; this requires that some argument should be made about the satisfaction of the (crisp) semantic-based meanings of the basic representation even in the cases when the degree of confidence is neither zero nor one. In this deliverable, we will concentrate on the mapping construct of the representation language, which is the crucial construct involved in the alignment procedure.

This will be the role of Deliverable 2.2.5 to elaborate a better and concrete format for these alignments. The current structure will be sufficient for the purposes of the present deliverable.

Chapter 4

Use cases

In order to motivate the interest for ontology alignment in the context of the semantic web and semantic web services, we present here a number of use cases. Each case is presented through its context, the heterogeneity problems raised by the interoperation of knowledge resources in this context and illustrates the potential solution to these problems offered by alignment. They are not necessarily implemented use cases.

4.1 Agent communication

Agents are computer entities characterized by their autonomy and capacity of interaction. They are often divided in cognitive agents and reactive agents. Reactive agents implement a simple behavior and the strength of these agents is their capacity to let a global behavior emerge from the individual behavior of many such agents. Cognitive agents have a rather more elaborate behavior often characterized by the ability to pursue goals, to plan their achievement and to negociate with other agents in order to achieve their goals.

Reactive agents are often used in applications in which the solution is not known in advance and a large number of agents are used for covering a large search space. Cognitive agents are rather used when a number of rational rules leading to a solution are known. In both case, the common ground is that these agents are autonomous and can adapt to their environment fairly easily. Software agents are often used for programming applications with agents that are intermediate between purely reactive and and fully cognitive.

The current models of cognitive agents considers their behavior as a set of beliefs, desires and intentions symbolically expressed and use speech-act inspired languages for interacting with each others. These languages determine the "enveloppe" of the messages and enable agents to position them within a particular interaction contexts. But they do not specify the actual content of the message which can be expressed by various content languages. In order to help them, currrent standards for expressing these messages provides slots for declaring the content language and the ontology used.

As a consequence, when two autonomous and independently designed agents meet, they have the possibility to exchange messages but little chance to understand each others if they do not share the same content language and ontology. It is thus necessary to provide the opportunity for these agents to align their ontologies in order to either translate their messages or integrate bridge axioms in their own models.

One solution to this problem would be to have an ontology alignment protocol able to be interleaved with any other agent interaction protocol and which could be triggered upon receiving a message expressed in an alien ontology. Such a protocol would allow communication with tier alignment services in order to:

- browse alignment libraries;
- ask for the (partial) alignment of some ontology pair;
- ask for the translation of a message given an alignment;
- ask for a translation program, bridge axioms or view definition given an alignement;
- ask for the completion of a partial alignment.

In addition, it should allow an agent to ask for agreement on some returned alignment or translate a message expressed with regard to some private ontology in a public one.

In such a case, the dialog between two agents could become:

agent1 sends a message m expressed with a private ontology o;
agent2 asks for a translation of m with regard to a public ontology o';
agent1 sends m' expressed with regard to o';
agent2 sends to **align** a request to align o' with ontology o'' with regard to message m;
align replies by partial alignment a;
agent2 asks **align** to translate m' thanks to alignment a;
align replies by m";
agent2 asks **align** to complete alignment a with terms in messge n;
align replies with alignment a';
agent2 asks align a translation program made from alignment a';
align replies with program p;
agent2 applies p to n and send the result as an answer to **agent1**.

4.2 Emergent Semantics

In what follows, we summarize the status of a collaborative effort on the development of the notion of "emergent semantics", which has been initiated by the IFIP 2.6 Working Group on Data Semantics. This summary is based on [Aberer *et al.*, 2004b] and [Aberer *et al.*, 2004a].

This approach is motivated by the belief that global semantic interoperability emerges from large numbers of purely local, pair-wise interactions. "Semantic interoperability is viewed as an emergent phenomenon constructed incrementally, and its state at any given point in time depends on the frequency, the quality and the efficiency with which negotiations can be conducted to reach agreements on common interpretations within the context of a given task". This type of semantic interoperability is called "emergent semantics".

The key principles of this approach are:

- Agreements as a Semantic Handshake Protocol. Emergent semantics ("Dynamic ontologies") can be established on the bases of mutually accepted propositions between the interacting agents. The quality of "emerged semantics" depends on the strength of the agreed propositions, and their trustworthiness.

- Agreements emerge from negotiations. Information environments are assumed dynamic. Thus, interactions between agents are necessary to identify and resolve semantic conflicts, and to verify whether a consensus leads to the expected actions. Interactions are message exchanges or references to distributed information resources.

– Agreements emerge from local interactions. Emergent semantics are assumed to be established incrementally, based on local agreements. Global agreements are obtained through aggregations of local agreements.

This approach is currently active in the area of peer-to-peer data management and integration, where local schema mappings are introduced in order to enable semantic interoperability. Local schema mappings can be seen as the local communication mechanisms for establishing consensus on the interpretation of data.

While the Semantic Web approach uses ontologies for obtaining semantic interoperability, the ambition of the emergent semantics approach is to obtain such interoperability in a more scalable and decentralized fashion, without necessarily using ontologies.

4.3 Web service integration

We understand web service discovery as the process of finding web services that fulfil a given requester goal. Both the requester goal and the service capability i.e. requested functionality and provided functionality are defined declaratively and in a machine-processable way.

Both the goal and the capability will be described using one or more domain-specific ontologies. Here we can find two different problems:

1. The descriptions of the capability or the goal use several domain ontologies that have to be mediated, as conflicts can arise, either from the conceptual model or from the ontology language used.

2. The capability and the goal are expressed using ontologies that describe a common domain, but still the ontologies used for the capability are different from the ones used for the goal. In this case, the discovery process still has to find suitable services despite the use of different terminologies for the goal and the capability.

Using the ontologies defined in [1] for train connections, locations, purchase orders and date and time, we illustrate here concrete problems for 1) and 2)

1a Use of different ontology languages The train connections ontology (Listing 1 in note 1) imports an ontology for persons described in OWL[2] and two ontologies described in WSML (F-Logic) for locations, and date and time. In the first case, a simple import is not possible as the ontologies are described in different languages (WSML and OWL). Therefore, the heterogeneity of ontology languages has to be overcome.

1b Changes on the conceptual model The capability described of a web service selling train tickets make use of the train connections ontology and the purchase order ontology (Listing 3 in note 1). This capability wants to express that the items of the trade it establishes are train trips. However, the range of the “items” property of the “trade” concept in the purchase order ontology is the concept “product”. For that reason, when importing the ontologies the capability has to say that “train trip” is a subconcept of “product”.

[1]http://www.wsmo.org/2004/d3/d3.2/

[2]http://daml.umbc.edu/ontologies/ittalks/person

1c Using different ontologies for the same domain A capability of a service selling flight tickets can use an airport codes ontology[3] which defines “city” as a property of the “AirportCode” concept, and the same time use a different ontology for countries[4], in which “country” is a concept and not a property. These two ontologies have to be used in a consistent way, resolving possible conflicts.

2a The goal is described using the locations ontology (Listing 4 in note 1) which defines the concept “Country” as an extension of the concept “Country” in the CIA factbook[5]. The service capability is described using an airport code resource[6], and it uses the “Country” property of the “AirportCode” concept. If the goal says that a service providing a flight from the country “Austria” to the country “Spain” is required, and the capability offers flights from airports with the property “Country” with value “Austria” to airports with the property “Country” with value “Spain”, a match have to be established regardless of the different conceptual model.

The heterogeneity problems above must be solved in order to enable the reuse of (possibly conflicting) ontologies for goal and capability descriptions, and in order to enable the match of goals and capabilities providing compatible functionalities but described using heterogeneous ontologies.

4.4 Ontology-driven data integration: The fund finder application

Our use case is about migrating relational database content to the semantic web allowing the integration of information from multiple and heterogeneous sources. Typically the input to this kind of problem is a set of n databases, each of them containing the data to be migrated and an ontology to be populated with instances extracted from the databases.

The important idea behind our approach is that mappings between the database SQL schema elements (tables, columns and keys) and the corresponding concepts, relations and attributes of the ontology will be defined declaratively in a mapping document. This mapping document will be the input of a processor charged of carrying out the effective migration in an automatic way. The fact of defining these mappings declaratively will make our solution domain independent and reusable. Another important aspect in our approach is that it uses the database and ontology as they are, we do not create the ontology from the database schema, that’s why some complex mapping situations may arise. Basically, the level of complexity of the mappings to be defined will depend on the level of similarity of the ontology’s model and the database schema. Normally, one of them will be richer, more generic or specific, better structured, etc., than the other.

We have created R2O, an extensible and fully declarative language to describe mappings between relational database schemas and ontologies. R2O is intended to be expressive enough to describe the semantics of these mappings and is proposed as a DBMS independent high level language that can work with any database implementing the SQL standard. The R2O language allows the definition of explicit correspondences between components of two models. A basic approach to the ideas underlying R2O is showed by the following expression:

[3]http://www.daml.ri.cmu.edu/ont/AirportCodes.daml
[4]http://www.daml.org/2003/09/factbook/factbook-ont
[5]http://www.daml.org/2003/09/factbook/factbook-ont
[6]http://www.daml.ri.cmu.edu/ont/AirportCodes.daml

$$OntologyComponent_i = Transformation(DatabaseComponent_j, DatabaseComponent_k?)$$

Where $OntologyComponent_i$ is any concept, attribute or relation in the target ontology and $DatabaseComponent_j$ is any database table or column.

A mapping between a database schema and an ontology can then be defined as a set of basic mapping expressions or mapping elements between components in both models like the one showed before. Details on the language and the use case can be found at [Barrasa *et al.*, 2003].

This approach has been implemented and tested with the Fund Finder application[7] which has been developed in the context of the ESPERONTO project. In our particular case, the database we want to migrate (FISUB) contains incentives and funds provided by the Catalan and Spanish Governments and by the European Union, for companies or entrepreneurs located in the Spanish region of Catalonia. It contains more than 300 registers that are updated manually on a daily basis. The reason why we want to migrate these contents to the Semantic Web is to be able to aggregate to them information from other web resources related to funding in the European Union and to allow web users to ask intelligent queries about funding resources according to some parameters like their profile, to look for complementary ones, to check compatibilities and incompatibilities between types of funding, and so on.

4.5 Catalog matching

Many e-Commerce application are based on the publication of electronic catalogs which describe the goods on sale and allow customers to select the goods they need. However, a very important obstacle to the success of distributed e-Commerce applications is the problem of interoperating different catalogs. Indeed, many systems require participant parties to perform very costly operations on their local catalogs to enter into the system, and this is a tremendous barrier for newcomers. Some typical instances of this situation are:

e-Marketplaces electronic malls where different sellers provide their goods in a common environment. The problem is that each provider typically owns a local catalog, in which goods are organized according to criteria that suit its internal business processes. However, to take part in the marketplace, providers should translate their catalogs into a common catalog, which will be presented to users as a single access point to what is sold in the marketplace. Notice that, in principle, this translation is required for each marketplace in which a company is involved, which means that a potentially very high number of catalogs should be maintained at the same time by each company. This is considered one of the strong barriers against the success of eMarketplaces.

products and services reclassification there are some efforts in trying to harmonize catalogs through the adoption of standard classification structures. Well-known examples are the United Nations Standard Products and Services Code - UNSPSC[8] and eCL@ss[9]. The problem is that adopting one of these standards would require companies to translate their catalogs into the standard one, and this can be a very costly activity, which would impact the management of their internal business processes.

[7]http://www.esperonto.net/fundfinder

[8]http://www.unspsc.org/

[9]http://www.eclass-online.com/

aggregation of buyers' needs in many e-Procurement scenarios, it would be important for buyers to aggregate their product demand to get some advantage in terms of supply conditions or buying power. To support the aggregation process, the system should be able to identify groups of buyers interested in a similar category of product, and possibly to suggest buyers how to modify some requested features to increase the advantages of aggregation. This should be possible without presupposing that differnet organizations share the same classification system.

The scenarios above (and many others) would be much more appealing (or would simply become viable) if we could provide means for aligning catalogs through rich mappings which allow the system to compare demand and offer. Moreover, in some applications (e.g. aggregation of buyers' needs), this mapping process should be (semi or fully) automatic, and runtime.

Notice that catalogs are a significant challenge for alignment, as catalogs are not simple concept hierarchies (classifications), but classifications in which classes may have multiple attributes. For example, a shirt has a size, a color and a price. Therefore, mappings should align not only the concept of shirt with an equivalent concept on another catalog, but should take into account the alignement of attribute names and values.

4.6 Information retrieval from heterogeneous multimedia databases

Digital archiving of multimedia content including video, audio, still images and various types of multimedia documents has been recognized by content holding organizations as a mature choice for the preservation, preview and partial distribution of their assets. The advances in computer, data networks and web technologies along with the success of standardization efforts of MPEG and JPEG boosted the movement of the archives towards the conversion of their fragile and manually indexed material to digital, web accessible data. By the end of last century the question was not on whether digital multimedia archives are technically and economically viable, but rather on how they would be efficient and informative. In this framework, different scientific fields such as database management systems, multimedia information processing, artificial intelligence, etc., have observed a close cooperation with each other during the last few years. The attempt has been to develop intelligent and efficient information retrieval systems, enabling the user to access vast amounts of heterogeneous multimedia information, stored in different sites and archives. Lately, using the advanced technologies of multimedia standardization, large databases and semantic web, access to heterogeneous multimedia archives is done throughout the following steps: - construction of large multimedia databases providing the raw multimedia information (images, video, audio, text, etc.) and its annotation (syntactic and semantic description) given in standardized (MPEG-4, MPEG-7, MPEG-21 etc) or not standardized metadata, - construction of ontologies providing the semantics of the above metadata, - construction of mediators (advanced search engines) unifying the multimedia annotation and the user access to heterogeneous multimedia content. Ontology and data alignment will play an important role in the above framework. It has become clear among the research community dealing with content-based multimedia data retrieval and new emerging related standards, that the results to be obtained will be ineffective, unless major focus will be given to the semantic unification of the heterogeneous content annotations. Algorithms and tools that find the correspondence between the semantics of metadata stored in the database of each archive will be based on the alignment of the ontologies providing these semantics.

4.7 P2P information sharing

Peer-to-Peer (P2P) information sharing systems have recently received a lot of attention both from the academia and industry. P2P file sharing systems already have a variety of implementations and are widely used on the Web (for instance, Kazaa and Morpheus have more than 450 million of downloads (see http://www.download.com, March 2004). Some of these applications provide a simple schema in order to describe file contents (e.g., Kazaa, Napster), which is shared by all parties and can not be changed locally at some party. Other P2P file sharing applications do not provide any schema at all (e.g., Gnutella), and encapsulate semantic information into the file names.

P2P information sharing systems which use rather complex structures to describe data (e.g., database schemas, ontologies in distributed knowledge management) have been largely studied in the academia but, to our knowledge, did not go far beyond prototypical test beds. The main underlying idea of these approaches is that peers define pair-wise mappings between their schemas (or ontologies) and use these mappings in order to propagate requests and data. Most of the above mentioned works are based on the assumption that the mappings between peer schemas are already defined a priori, and they do not consider how the mappings are actually created.

A real life example for P2P databases could be the real estate agents example. Agents coordinate their databases in exchanging real estate information with the goal of pushing sales. Different agents may use different schemas to describe their data, so that they establish mappings between their schemas to enable interoperation. Since they travel to their customers (who may want to sell or, instead, to buy), they always carry relevant data with them. When one is on the site of a customer, who wants to sell a house, the agent updates his database and makes this data available for other agents. Or, when an agent talks with a potential buyer, and nothing from the agent's database satisfies the client, the agent may want to query other agents' databases to look for additional sale options.

In P2P settings assumptions that all parties agree on the same schema, or that all parties rely on one global schema (as in data integration) can not be made. Peers come and go, import multiple schemas into the system, and have a need to interoperate with other nodes at runtime. In this activity we see schema alignment as the main process to enable nodes' interoperation. Namely, when two peers "meet" on the network, they establish mappings between their schemas in a (semi) automatic alignment discovery process.

Automation of the schema alignment discovery process will create a great advance for the P2P information sharing systems on the Semantic Web. Peers will be able to bring to the system various schemas, "align" them to the schemas of other peers on the network with no (or minimal) user involvement, and exchange requests and data in a decentralized, collaborative manner. So far the heterogeneity problem has been the main obstacle for the P2P applications with "rich" heterogeneous schemas to flow into the Web infrastructure (as in the case of file sharing systems). But the advance of the Semantic Web technologies allows us to make an optimistic assumption that it will be the case in the nearest future.

4.8 GridVine: aligning schemas in overlay networks

P2P systems construct so-called *overlay networks* over the physical network. In principle, applications could use the services provided by the networking layer to locate their resources of interest. However, having a separate, application-specific overlay network has the advantage of supporting application-specific identifiers and semantic routing and offers the possibility to provide additional services for supporting network maintenance, authentication, trust, etc., all of which would be hard to integrate into and support at the networking layer. The introduction of overlay networks is probably the essential innovation of P2P systems.

In the GridVine use case we want to build a semantic overlay network which provides structured search and (semi-) automatic schema integration based on the semantic gossiping approach [Aberer *et al.*, 2003b] as described in Section 7.3.3. As a case study we will use the domain of image annotation. The setup we assume is as follows: People take pictures with digital cameras and want to share them with others via a P2P system. By default digital cameras store images under cryptic names not related to the image content and can add technical annotations, for example, the date and the time the photo was taken. In GridVine we want to allow the user to freely define application-specific schemas, for example, annotated photos with meaningful names, descriptions, or any other data they consider meaningful, without imposing a global schema. This assumption has been shown to be realistic since in P2P systems with millions of users and no authority that can enforce standards, it is impossible to impose global schemas. Additionally, a global schema may neglect useful user knowledge that could otherwise be provided.

Concretely, users will be able to define individual schemas in RDFS to describe their photo images. These schemas are indexed by the P2P system and thus can be found and retrieved by any participant via RDQL. If a user is interested to integrate his/her schema with the schema of another user he/she can provide a translation in OWL. Thus, a network of semantic translations among end-user schemas will emerge. Again these translations are indexed in the P2P system and can be retrieved by other participants which serves as the basis to support semantic gossiping and schema integration.

In resolving translation links and assessing their quality we will investigate iterative and recursive approaches. In iterative resolution the peer issuing the RDF query will try to find all translations links itself: The peer issues a query for the translation of a certain concept. Upon finding a translation, it translates the original query using the found translation (Query') and issues a search for the transformed query. Furthermore, the gossiping peer will issue a query for a translation of the translation (Predicate'). This continues until no more translations are available or the transformed query is considered as being too different from the original query following syntactic and semantic similarity values [Aberer *et al.*, 2003b].

In recursive resolution, the issuing peer tries to resolve the translation by delegating rather than doing it itself: First, it looks for translations of the concepts used in the query and translates the query upon finding a translation. The transformed query is issued and results for the query will be returned to the issuer of the query. The receiver of the transformed query will follow the exact same procedure, and so on recursively.

In the case study we will investigate the applicability of our semantic gossiping approach in a practical setting. We will implement GridVine on top ouf our P-Grid P2P system [Aberer *et al.*, 2003a] and perform experiments to assess the achievable quality of schema integration and to obtain performance and scalability figures of the approach.

4.9 Personnal information delivery

These days Internet radio is becoming reality. A number of audio feeds are now available on the web. Philips is already working on a stand alone media station supporting internet radio streams.

At the moment internet radio is not really different from conventional radio concerning the limited ability to influence the content of the radio program. The idea of smart internet radio is now to exploit the possibilities of web technology, in order to provide internet radio that is customized to match the users interests by combining different audio streams and selecting the most appropriate stream. The idea is that the based on model of the users interests an appropriate audio stream is selected and played.

Besides more technical problems like dealing with different audio formats and timing of audio sequences, there are some conceptual problems to be addressed. These problems originate from the need to compare user interests with metadata provided with the audio streams. Besides information about artists, songs or artists are normally assigned to different musical genres. In principle, information about genres can be used to select songs the user is likely to be interested in. The problem with this approach is that there is neither a universal agreement on a fixed set of genres nor is there an agreement on the correct classification of songs into genres. The idea is now to use semantic web technologies, in particular explicit representation of terminologies and their intended meanings. While hierarchies of genres can often be found on the web sites of audio content providers such as MusicMoz[10] or AllMusic[11] the classification of music genres that reflects the opinion of a user can often be obtained from the users file systems. The corresponding integration problem is two-fold:

– The categorizations of audio sources have to be matched against the users interests reflected in his or her personal categorization of music
– The categorizations of different information providers have to be compared in order to be able to use matches and user feedback on a single source to draw conclusions about other sources

The specific problem of an alignment of genre information lies in the fact that there is no agreement about the right categorization of artists or even songs into genres. This categorization can be different from person to person. On the other hand, linguistic approaches for comparing categories are doomed to fail due to the artificial nature of genre names that do not have a connection to the standard semantics of natural language.

4.10 Vertical Publishing in Life Science Domain

Innovative research institutes rely on the availability of complete and accurate information about new research and development, and it is the business of information providers such as Elsevier to provide the required information in a cost-effective way. It is very likely that the semantic web will make an important contribution to this effort, since it facilitates access to an unprecedented quantity of data. However, with the unremitting growth of scientific information, integrating access to all this information remains a significant problem, not least because of the heterogeneity of the information sources involved – sources which may use different syntactic standards (syntactic heterogeneity), organize information in very different ways (structural heterogeneity) and even use different terminologies to refer to the same information (semantic heterogeneity). The ability to address these different kinds of heterogeneity is the key to integrated access.

[10]http://musicmoz.org/Styles/
[11]http://www.allmusic.com/mus_Styles.html

Thesauri have already proven to be a core technology to effective information access as they provide controlled vocabularies for indexing information, and thereby help to overcome some of the problems of free-text search by relating and grouping relevant terms in a specific domain. Two examples of thesauri in the life sciences are MeSH[12], which is produced by the U.S. National Library of Medicine (NLM) and Elsevier's life science thesaurus EMTREE[13]. In the medical area a lot of work has been done on the definition and standardization of terminologies. The result of these efforts is a large number of medical thesauri such as MeSH. The complexity of the terminologies used in medicine and the strong need for quality control has also lead to the development of ontologies that feature complex concept definition. Some of these ontologies are available in OWL and can be seen as the first OWL applications that have a use in real life applications. The need for an integration of exiting thesauri and ontologies has been widely recognized in the medical area leading to a number of efforts for defining standardized terminologies. It is, however, also acknowledged by the literature, that the creation of a single universal terminology for the medical domain is neither possible nor beneficial, because different tasks and viewpoints require different, often incompatible conceptual choices. As a result a number of communities of practice have been evolved that commit to one of the proposed standards. This situation demands for a weak for of integration, also referred to as alignment in order to be able to exchange information between the different communities.

This implies that terminology alignment is important for new publishing models known as "vertical publishing". Different from the traditional model where information providers sell predefined source information (e.g., in terms of a medical journal) in vertical publishing, the information provider sells information about a certain topic independent from the source (e.g., by retrieving relevant articles from different journals). As relevant information may be provided by different communities, the respective terminologies have to be aligned and compared with the terminology of the customer in order to provide all relevant information.

[12]http://www.nlm.nih.gov/mesh/meshhome.html

[13]http://www.elsevier.com/homepage/sah/spd/site/

Chapter 5

Ontology alignment process

Mappings are the basic building blocks of ontology alignment. The goal of this section is to provide a precise definition of what the alignment process is in general, and what are its main dimensions.

Determining these dimensions is very important for characterizing what known or yet to be invented alignment algorithm does and then in which situation it is adapted. It should also be very useful in designing benchmark tests and comparing similar algorithms.

5.1 Characterization of the alignment process

The alignment process simply consists of generating an alignment (A') from a pair of ontologies (o and o'). However, there are various other parameters which can extend the definition of the alignment process. These are namely, the use of an input alignment (A) which is to be completed by the process, the alignment methods parameters (which can be weigths for instance) and some external resources used by the alignment process (which can be general-purpose resources not made for the case under consideration, e.g., lexicons, databases). This process can be defined as follow:

Definition 1 (Alignment process)**.** *The alignment process can be seen as a function f which, from a pair of ontologies o and o' to align, an input alignment A, a set of parameters p, a set oracles and resources r, returns a new alignment A' between these ontologies:*

$$A' = f(o, o', A, p, r)$$

This can be represented as in Figure 5.1.

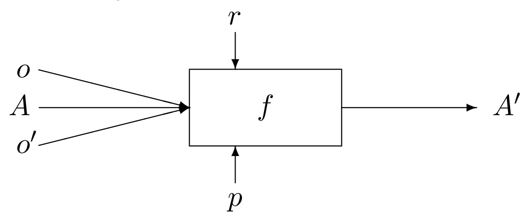


Figure 5.1: The alignment process.

Moreover, it can be useful to specifically consider the alignment of many ontologies within the same process. We call this multi-alignment.

Definition 2 (multi-alignment process)**.** *The multi-alignment process can be seen as a function f which, from a set of ontologies to align $\{o_1, \dots o_n\}$, an input multi-alignment A, a set of parameters p, a set oracles and resources r, returns a new alignment A' between these ontologies:*

$$A' = f(o_1, \dots o_n, A, p, r)$$

5.2 Dimensions of an alignment process

Beside the general scheme presented above, there are many restrictions that can be put on this alignment process. These restrictions are useful for either constraining the alignment algorithm to deliver a particular kind of alignment (e.g., 1-1 preserving consequences) or to choose an algorithm adapted to the required constraints.

These dimensions affect most of the components of the above alignment definition:

Input ontologies (o**,** o'**)** Input ontologies can be applied various constraints:

- **heterogeneity** of the input languages: are they described in the same knowledge representation languages? This corresponds to asking for the non emptyness of the syntactic component of the resulting alignment.
- **languages:** what are the languages of the ontologies (especially in case of homogeneous languages)? Example of languages are KIF, OWL, RDFS, UML, F-Logic, etc.
- **number:** is this an alignment or a multi-alignment?

Input alignment (A**)** The input alignment:

- **complete/update:** Is the alignment process required to complete an existing alignment? (i.e., is A non empty).
- **multiplicity** : How many entities of one ontology can correspond to one entity of the others? Usual notations are 1:1, 1:m, n:1 or n:m. We prefer to note if the mapping is injective, surjective and total or partial on both side. We then end up with more alignment arities (noted with, 1 for injective and total, ? for injective, + for total and * for none and each sign concerning one mapping and its converse): ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:*. These assertions could be provided as input (or constraint) for the alignment algorithm or be provided as a result by the same algorithm.

Parameters (p**,** r**)**

- **oracles/resources** Are oracle authorized? If so, which ones (the answer can be any)? Is human input authorized?
- **training** Can training be performed on a sample?
- **proper parameters** Are some parameter necessary? And what are they? This point is quite important when a method is very sensitive the variation of parameters. A good tuning of these must be available.

Output alignment (A')

- **multiplicity** The multiplicity of the output alignment is similar to that of the input alignment (see above).
- **justification** Is a justification of the results provided?
- **relations** Should the relations involved in the correspondences be only equivalence relations or could they be more complex?
- **strictness** Can the result be expressed with trust-degrees different than $\top$ and $\bot$ or should they be strictified before?

Alignment process (f) The alignment process itself can be constrained:

- **resource constraints** Is there a maximal amount of time or space available for computing the alignment?
- **Language restrictions** Is the mapping scope limited to some kind of entities (e.g., only T-box, only classes)?
- **Property** Must some property be true of the alignment? For instance, one might want that the alignment (as defined in the previous chapter be a conseqeunce of the combination of the ontologies (i.e., $o, o' \models A'$) or that alignments preserve consequences (e.g., $\forall \phi, \phi' \in L, \phi \models \phi' \implies A'(\phi) \models A'(\phi')$) or that the initial alignment is preserved (i.e., $o, o', A' \models A$).

The purpose of the dimensions is the definition of the parameters and characteristics of expected behavior in benchmark and the comparison of algorithms and systems in deliverable D2.2.3.

5.3 Data alignment and integration

Data alignment and integration consists in merging data (and sometimes data streams, d and d') expressed in different ontologies (o and o'). For that purpose, the ontologies have to be aligned beforehand and the data integration can use this alignment. This is an example of combined off-line and on-line alignment.

It can be thought of as:

1. a first ontology alignment phase (f), possibly with an instance training set,
2. a data alignment phase (f') using the first alignment (A').

This is presented in Figure 5.2.

In this setting, the second phase benefits from the precompiling of the first alignment. Indeed, the second alignment process f' can be thought of as a compilation of the first alignment. This covers enough applications to deserve a separate threatment (e.g., for benchmarking).

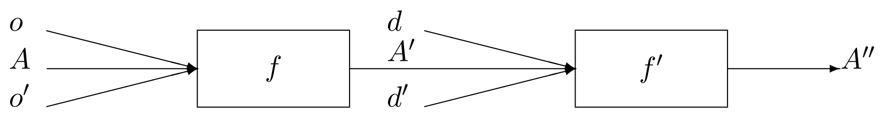


Figure 5.2: Data integration as another alignment process.

5.4 Conclusion

This document provided very general characterisation of the nature of alignments. They have first be considered under conceptual facets of the context in which alignments are needed (and thus what they align). Then a categorical framework has characterised alignments as the seed for the merge operation (itself defined as a push-out). Furthermore, global and distributed model-theoretic semantics have been given to the mappings composing alignments. Finally, the alignment process, by which alignments are produced has been proposed a definition covering most of the actual systems.

As could be expected from such a broad topic, the provided definitions are very abstract. We feel that they cover most of what have been implemented as alignment systems. Some of the definitions presented here provide however a very acute and fruitful view of what alignment are.

This work is very useful for designing tools and alignment algorithms and examining their properties. It will be used within the network for proposing a common alignment format that can be exchanged among a variety of tools (aligners, merger, transformers, etc.). It has been used as well for designing the alignment benchmarks of Deliverable 2.2.2.

These characterisations raise very interesting questions such as: is it possible to further characterise alignments or most specific alignments in the categorical framework? Can model theory account for more complex structure in the relations between distributed sites (for instance, by modelling the effort required by a site to acquire knowledge)? These questions are worth investigating for grounding further the work on ontology reconciliation in Knowledge web. We will devote some resources toward this.

Part II

State of the art

Chapter 6

Local methods

The main issue in aligning consist of finding to what entity or expression in one ontology corresponds another one in the other ontology. Here are presented the basic methods which enable to measure this correspondence at a local level, i.e., only comparing one element with another and not working at the global scale of ontologies.

Very often, this amounts to measuring a pair-wise similarity between entities (which can be as reduced as an equality predicate) and computing the best match between them, i.e., the one that minimizes the total disimilarity (or maximizes the similarity measure).

There are many different ways to compute such a dissimilarity with different methods designed in the context of data analysis, machine learning, language engineering, statistics or knowledge representation. Their condition of use depends of the objects to be compared, their context and sometimes the external semantics of these objects. Some of this context can be found in Figure 6.1 (From [Rahm and Bernstein, 2001] enhanced in [Giunchiglia and Shvaiko, 2003a; Shvaiko, 2004b] and [Euzenat and Valtchev, 2003]) which decomposes the set of methods along two perspectives: the kind of techniques (descending) and the kind of manipulated objects (ascending).

After providing base definitions about ontologies and measures (§6.1), the outline of the chapter follows the lower classification of Figure 6.1: it is decomposed in terminological (§6.2), structural (§6.3), extensional (§6.4) and semantic methods (§6.5). Their combination is explored in next chapter and their use in actual systems presented in Chapter 8.

6.1 Ontologies and measures

In order to fix the vocabulary used in this deliverable, some definitions are given here concerning ontologies and similarities. They are only presented for the sake of completeness and can be skipped without harm by the knowledgeable reader.

6.1.1 Ontology language

We will not give a definition of ontology but rather consider that an ontology is expressed in an ontology language and alignment methods must work in function of these languages and their features. There are a large variety of languages for expressing ontologies (see [Staab and Studer, 2004]). It is not the purpose of this deliverable to present them all or to commit to one particular

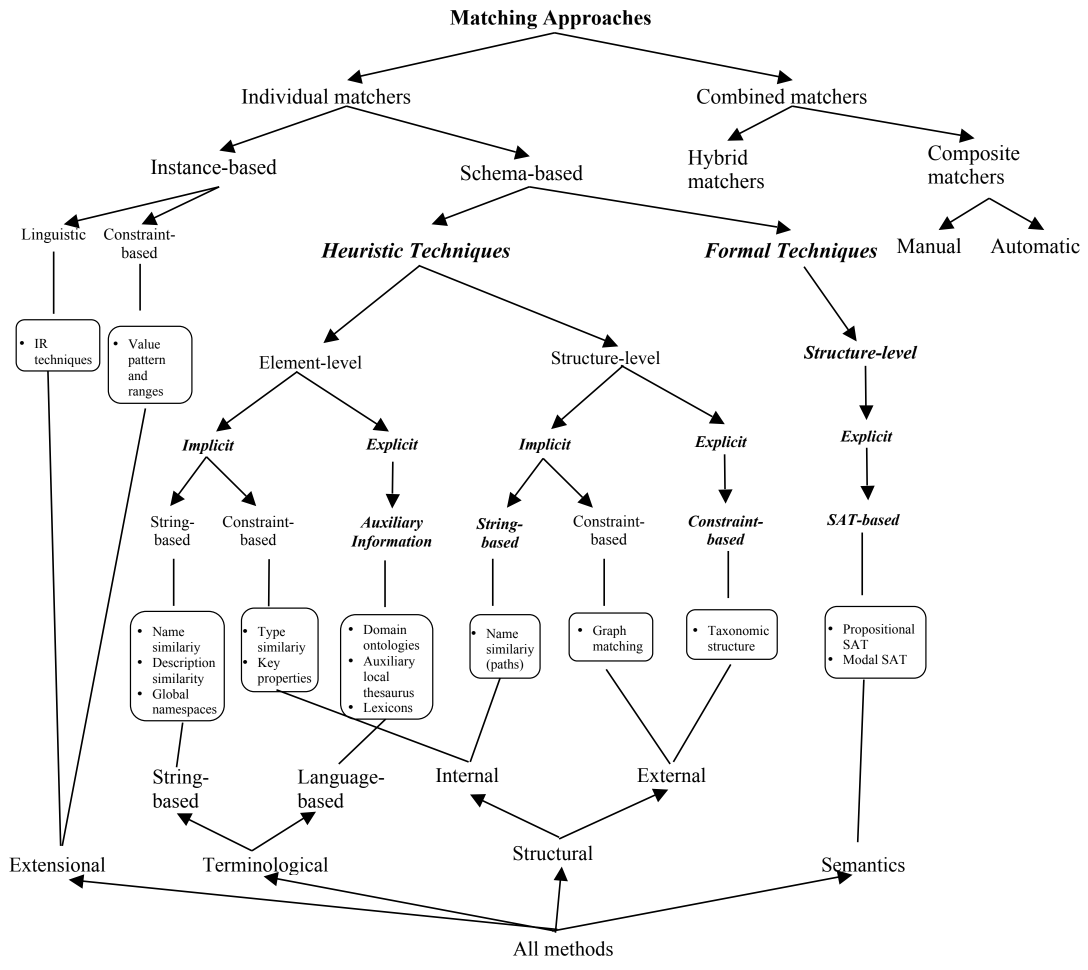


Figure 6.1: Classification of local methods.

language. Fortunatelly, they most often share the same kind of entities (with different names but comparable interpretation). So, we very shortly describe here what entities are found in an ontology languages. This is the goal of the local methods to assess the correspondence of the entities in these languages. These entities are mainly:

classes or concepts are the main entities of an ontology. They are interpreted as a set of individuals in the domain.

objects or instances are interpreted as particular individual of a domain;

relations are the ideal notion of a relation independently to why it applies (e.g., the name relation in itself), they are interpreted as a subset of the products of the domain.

properties are the relations precisely applied to a class (the name of a man);

property instances are the relations applied to precise objects (the name of this individual)

datatypes are a particular part of the domain which specifies values (as opposed to individuals), values do not have identities;

datavalues are simple values.

property restrictions are the constraints applying to properties (they restrict the interpretation of the property for a class).

These entities are linked by various kinds of relationships that are also very common:

specialization (or subsumption) between two classes or two properties (interpreted as inclusion of the interpretations);

exclusion between two classes or two properties (interpreted as the exclusion of their interpretations, i.e., their intersection is empty);

instanciation (or typing) between objects and classes, property instances and properties, values and datatypes (which is interpreted as membership);

attribution between classes and properties, objects and property instances;

restriction expressing the restriction on a property in a class;

assignment of a property in an individual

Most of these features are found in modern ontology languages (e.g., OWL). Other kinds of constructs exist such as arbitrary formulas (or axioms). They are not discussed here.

6.1.2 Similarity and other measures

There are many ways to assess the similarity between two entities. The most common way amounts to defining a measure of this similarity. We present the characteristics which can be asked from these measures.

Definition 3 (Similarity)**.** *A similarity $\sigma : O \times O \to \mathbb{R}$ is a function from a pair of entities to a real number expressing the similarity between two objects such that:*

$$\forall x, y \in O, \sigma(x, y) \geq 0 \qquad \textit{(positiveness)}$$
$$\forall x \in O, \forall y, z \in O, \sigma(x, x) \geq \sigma(y, z) \qquad \textit{(maximality)}$$
$$\forall x, y \in O, \sigma(x, y) = \sigma(y, x) \qquad \textit{(symmetry)}$$

The dissimilarity is a dual operation:

Definition 4 (Dissimilarity)**.** *Given a set O of entities, a dissimilarity $\delta : O \times O \to \mathbb{R}$ is a function from a pair of entities to a real number such that:*

$$\forall x, y \in O, \delta(x, y) \geq 0 \qquad \textit{(positiveness)}$$
$$\forall x \in O, \delta(x, x) = 0 \qquad \textit{(minimality)}$$
$$\forall x, y \in O, \delta(x, y) = \delta(y, x) \qquad \textit{(symmetry)}$$

Some authors consider a "non-symmetric (dis)similarity", e.g. [Tverski, 1977], we will then use the term non-symmetric measure. There are more constraining notions of dissimilarity such as distance and ultrametrics.

Definition 5 (Distance)**.** *A distance (or metrics) $\delta : O \times O \to \mathbb{R}$ is a dissimilarity function satisfying the definiteness and triangular inequality:*

$$\forall x, y \in O, \delta(x, y) = 0 \textit{ iff } x = y \qquad \textit{(definiteness)}$$
$$\forall x, y, z \in O, \delta(x, y) + \delta(y, z) \geq \delta(x, z) \qquad (triangular\ inequality)$$

Definition 6 (Ultrametrics)**.** *Given a set O of entities, an ultrametrics is a metrics such that:*

$$\forall x, y, z \in O, \delta(x, y) \leq max(d(x, z), d(y, z)) \qquad \textit{(ultrametric inequality)}$$

Very often, the measures are normalized especially if measure of the similarity of different kind of entities must be compared. Reducing each value to the same scale (i.e., proportionnaly to the size of the considered space) is the common way to normalize.

Definition 7 (Normalized (dis)similarity)**.** *A (dis)similarity is said to be* normalized *if it ranges over the unit interval of real numbers* $[0\ 1]$. *A normalized version of a (dis)similarity σ (resp. δ) will be noted $\overline{\sigma}$ (resp. $\overline{\delta}$).*

It is easy to see that to any normalized similarity $\overline{\sigma}$ corresponds a normalized dissimilarity $\overline{\delta} = 1 - \overline{\sigma}$ and vice-versa.

In the remainder, we might consider only normalized measures. We will assume that a dissimilarity function between two entities must return some real number between 0 and 1.

6.2 Terminological methods

Terminological methods compare strings. They can be applied to the name, the label or the comments concerning entities to find those which are similar. This can be used for comparing class names and/or URI.

Throughout this section, the set $\mathbb{S}$ will represent the set of strings, i.e., the sequences of letters over an alphabet $\mathbb{L}$ (so, $\mathbb{S} = \mathbb{L}*$). The empty string is noted ϵ, and $\forall s, t \in \mathbb{S}$, $\forall c \in \mathbb{L}$ $s + t$ is the concatenation of the strings s and t (which will not be further defined). $|s|$ will be the length of the string s (i.e., the numbers of characters it contains). $s[i]$ for $i \in [1\ |s|]$ will be the letter in position i of s.

A string s is the substring of another t, if there exists two strings s' and s'' such that $s' + s + s'' = t$ (this is noted $s \in t$). Two string are equal ($s = t$) if and only if $s \in t$ and $t \in s$. The number of occurence of s in t (noted $s\#t$) is the number of distinct pairs s', s'' such that $s' + s + s'' = t$.

In the field of terminology, the relation between terms and concepts tends to be distinctly multivocal [Maynard, 1999], i.e. terms can refer to more than one concept, and a single term can have many variants, all related to a single concept. Although this is strictly prohibited by ISO Standard 704 (Principles and Methods for Terminology, 1987), modern terminological theory accepts that this is neither practical nor even desirable and rejects the rather narrow prescriptive view of the past in favour of communicative theories requiring different forms in different situational needs [Sager, 1990]. Such problems hold equally for any ontology – domain-specific or not — and for any kind of instance, not just for specialised terms. URIs used as names in the semantic web do not require that the names be unique, although identical names must refer to the same entity. So two objects may be identical, but have different URIs

If this problem exists even within a single ontology, it is increased tenfold when two or more ontologies are merged. There is no way that the use of terms can be expected to be consistent across merged ontologies, and different parts of the merged ontology may have conflucting or ambiguous elements within them, not just for concepts and instances, but also for relations.

There are two main categories of methods for comparing terms depending on their consideration of character strings only (§ 6.2.1) or if they use some linguistic knowledge (§ 6.2.2).

6.2.1 String-based methods

String-based methods take advantage of the structure of the string (as a sequence of letter). String-based methods will typically find as similar classes `Match` and `match`, but not `alignment`.

There are many ways to compare strings depending of the way the string is seen (as an exact sequence of letters, an erroneous sequence of letters, a set of letters, a set of words…). The most frequent are presented below. [Cohen *et al.*, 2003] compares various string-matching techniques, from distance like functions to token-based distance functions.

Normalization

Before comparing strictly strings which have meaning in (occidental) natural language, there are a number of normalization procedures that help improving the results of subsequent comparison:

Case normalisation consists of converting each alphabetic character in the strings in their downcase counterpart;

Diacritics suppression consists in replacing characters with diacritic signs with their most frequent replacement (e.g., replacing *Montréal* with *Montreal*);

Blank normalisation consists of normalising all blank characters (blank, tabulation, carriage return, our sequence of theses) into a single blank character;

Link stripping consists of normalizing some links between words (like replacing apostrophes and blank underline into dashes;

Digit suppression consists of suppressing digits (to be used with care, there are chemical names containing digits);

Punctuation elimination is useful when only words are considered and not sentences;

Stopword elimination eliminates words that can be found in a list (usually like, “to”, “a”…). This is usually used for comparing long texts.

String equality

String equality returns 0 if the string are not the same and 1 if they are the same.

Definition 8 (String equality)**.** *String equality is a similarity* $\sigma : \mathbb{S}\times\mathbb{S} \rightarrow [0,1]$ *such that* $\forall x, y \in \mathbb{S}$*,* $\sigma(x,x) = 1$ *and* $if x \neq y, \sigma(x,y) = 0$*.*

It can be performed after some syntactic normalisation of the string (e.g., downcasing, encoding conversion, accent normalisation).

This measure does not tell how strings are different. A more immediate way of comparing two strings is the Hamming distance which counts the number of positions in which the two strings differ (we give here the version normalised by the length of the largest one).

Definition 9 (Hamming distance)**.** *The Hamming distance is a dissimilarity* $\delta : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$ *such that:*

$$\delta(s,t) = \frac{(\sum_{i=1}^{min(|s|,|t|)} s[i] \neq t[i]) + ||s| - |t||}{max(|s|,|t|)}$$

Substring test

A number of variations can be obtained from the string equality such as considering that strings are very similar when one is a substring of another:

Definition 10 (Substring test)**.** *Substring test is a similarity* $\sigma : \mathbb{S}\times\mathbb{S} \rightarrow [0,1]$ *such that* $\forall x, y \in \mathbb{S}$*, if there exist* $p, s \in \mathbb{S}$ *such that* $x = p + y + s$ *or* $y = p + x + s$*, then* $\sigma(x, y) = 1$*, otherwise* $\sigma(x, y) = 0$.

This is obviously a similarity. This measure can be refined in a substring similarity which measures the ratio of the common subpart between two strings.

Definition 11 (Substring similarity)**.** *Substring similarity is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$ *such that* $\forall x, y \in \mathbb{S}$*, let* t *be the largest common substring of* x *and* y*,* $\sigma(x, y) = \frac{2|t|}{|x|+|y|}$.

It is easy to see that this measure is indeed a similarity. One could also consider a subsequence similarity as well. This definition can be used for building function based on the largest common prefix or largest common suffix.

The N-gram distance is also well used in comparing strings with some robustness:

Definition 12 (N-gram distance)**.** *Let* $ngram(s, n)$ *be the set of substrings of* s *(augmented with* $n - 1$ *irrelevant characters at the beginning and the end) of length* n*, the* n*-gram distance is a dissimilarity* $\delta : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}$ *such that:*

$$\delta(s, t) = |ngram(s, n) \cap ngram(t, n)|$$

The normalized version of this function is:

$$\overline{\delta}(s, t) = \frac{|ngram(s, n) \cap ngram(t, n)|}{n * min(|s|, |t|)}$$

This function is quite efficient when characters are only missing.

Edit distance

Generally speaking, an edit distance between two objects, is the minimal cost of operations to apply to one of the object for obtaining the other. The edit distance on strings (as known as Levenshtein distance) is the minimum number of insertions, deletions, and substitutions of characters required to transform one string into the other. Each operation much be assigned a cost.

Definition 13 (Edit distance)**.** *Given a set* Op *of string operations (*$op : \mathbb{S} \rightarrow \mathbb{S}$*), and a cost function* $w : Op \rightarrow \mathbb{R}$*, such that for any pair of strings there exist a sequence of operations which transforms the first one into the second one (and vice versa), the edit distance is a dissimilarity* $\delta : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$ *such that* $\delta(s, t)$*, is the cost of the less costly sequence of operations which transform* s *in* t*.*

$$\delta(s, t) = min_{(op_i)_I;op_n(\ldots op_1(s))=t}(\sum_{i\in I} w_{op_i})$$

In string edit distance, the operations usually considered are insertion of a character $ins(c, i)$, replacement of a character by another $sub(c, c', i)$ and deletion of a character $del(i, c)$. It can be easily checked that these operations are such that $ins(c, i) = del(i, c)^{-1}$ and $sub(c, c', i) = sub(c', c, i)^{-1}$. Moreover, it can be proven that the edit distance is indeed a distance if $\forall op \in Op, w_{op} = w_{op^{-1}}$.

The Levenstein distance is the edit distance with all costs to 1. The Needleman-Wunch distance is the edit distance with a higher costs for ins and del. Monger-Elkan distance function which has particular cost parameters for operations, scaling to the interval [0 1].

A roughly similar metric, but not based on an edit distance model, the Jaro-Winkler metric which is based on the number and order of the common characters between two strings is also compared.

Definition 14 (Jaro similarity)**.** *The Jaro similarity is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \longrightarrow [0, 1]$

$$\sigma(s, t) = \frac{1}{3}.(\frac{com(s, t)}{|s|} + \frac{com(t, s)}{|t|} + \frac{com(s, t) - transp(s, t)}{2 * com(s, t)})$$

in which

$$s[i] \in com(s, t) \textit{ iff } \exists j \in [i - (min(|s|, |t|)/4\; i + (min(|s|, |t|))]$$

and $transp(s, t)$ *are the element of* $com(s, t)$ *which are at different places in* s *and* t*.*

Token-based distances

Token-based distances are used for comparing pieces of texts rather than labels. They starts with a segmentation of the text into "token" (generally substrings of the initial strings) which are compared as (multi-)sets of such tokens instead of strings (these multi sets are also known as vectors in which each dimension corresponds to a term and the value corresponds to the number of term occurence in the string).

There exists several measures for this tasks which are based on comparing two sets of strings and are thus relevant of set comparison (see § 6.4). One can cite the Jaccard similarity, TF/IDF, cosine similarity, Jensen-Shannon distance and Fellegi and Sunter method extended to a token distance.

TF/IDF (Term frequency/Inverse document frequency) is usually not a measure of similarity: it assesses the relevance of a term to a document (and is used here to assess the relevance of a substring to a string by comparing the frequency of appearence of the string in the document with regard to its frequency in the whole corpus.

Definition 15 (Term frequency/Inverse document frequency)**.** *Given a corpus* C *of strings (usually documents), we define the following measures:*

$$\begin{aligned} \forall t \in \mathbb{S}, \forall s \in C, tf(t, s) &= s\#t && (term\ frequency) \\ \forall t \in \mathbb{S}, doc(t) &= |\{s \in C; t \in s\}| \\ \forall t \in \mathbb{S}, idf(t) &= log(\frac{|C|}{|doc(t)|}) && (inverse\ document\ frequency) \end{aligned}$$

Building a similarity from TF/IDF, amounts to computing them for the terms of both documents and aggregating and comparing the results. This is a special similarity measure since it is not intrinsic but dependent on a corpus (like most of the other cited methods).

Path comparison

Path difference consists in comparing not only the labels of objects but the sequence of labels of entities to which those bearing the label are related. A simple (and only) example is the one which concatenates all the names of the superclasses of a particular class before comparing it. So the result is dependent on the individual string comparison aggregated in some ways. The Comma system uses it [Do *et al.*, 2002].

Definition 16 (Path distance)**.** *Given two sequences of strings, $\langle s_i\rangle_{i=1}^n$ and $\langle s'_j\rangle_{j=1}^m$, their path distance is obtained by:*

$$\delta(\langle s_i\rangle_{i=1}^n, \langle s'_j\rangle_{j=1}^m) = \lambda.\delta'(s_1, s'_1) + (1-\lambda).\delta(\langle s_i\rangle_{i=1}^{n-1}, \langle s'_j\rangle_{i=1}^{m-1})$$

where

$$\delta(\langle\rangle, \langle s'_j\rangle_{j=1}^m) = \delta(\langle s_i\rangle_{i=1}^n, \langle\rangle) = 0$$

with δ' some of the other string or language based distances and $\lambda \in [0\ 1]$.

This measure is very dependant on the similarity between the last element of each paths. It takes into account the prefix, but it can only influence form some amount which decreases as their distance from the end of the sequence increases.

Another way to take these paths into account is simply to apply them a distance on sequences such as described in [Valtchev, 1999].

6.2.2 Language-based methods

Language-based methods rely on using Natural Language Processing (NLP) techniques to find associations between instances of concepts or classes. These methods may be either intrinsic (using the internal linguistic properties of the instances, such as morphological and syntactic properties) or extrinsic (requiring the use of external resources, e.g. lexicon-based and multilingual methods). Language-based methods essentially rely on the **expressive** and **productive** properties of natural language, which means that even technical terms can be expressed in many different ways without intrinsically altering their meaning [Maynard and Ananiadou, 1999]. This is usually referred to as **term variation**. Some of the most notable research in this area was carried out by the FASTR project [Jacquemin and Royaute, 1994; Jacquemin, 1996] which aimed to find alternative variants of terms for automatic indexing. The goal in this case was slightly different from our concerns with ontological alignment, but the methods are equally applicable since the idea is to identify whether two terms essentially refer to the same concept. In an ontology these terms may represent either instances of classes that we wish to match.

[Maynard and Ananiadou, 1999] distinguishes 3 main kinds of term variation: morphological, syntactic and semantic, although combinations of these are also possible (in particular, morphosyntactic variation is very common). We can add to this multilingual variation, i.e. where the term variant is expressed in a different language. These types can be subdivided further. Morphological variants can be divided into inflectional and derivational variants (or a combination of the two). Jacquemin and Royauté distinguish between 3 types of syntactic variants: coordination, permutation and insertion, and also define a separate category of morphosyntactic variants: a combination of (derivational) morphological and syntactic variants. However, all three main types of

Type	Subtype	Example
Morphological	Inflection	enzyme activities
	Derivation	enzymatic activity
	Inflectional-Derivational	enzymatic activities
Syntactic	Insertion	enzyme amidolytic activity
	Permutation	activity of enzyme
	Coordination	enzyme and bactericidal activity
Morphosyntactic	Derivation-Coordination	enzymatic and bactericidal activity
	Inflection-Permutation	activity of enzymes
Semantic		fermentation
Multilingual	French	activité d'enzyme

Table 6.1: Variants of the term *enzyme activity*.

variants can be combined in various ways. Table 6.1 depicts the types of variants and gives some examples of possible variants of the term *enzyme activity*.

Terminological methods allow to find the relation between these entities.

Intrinsic methods

These methods perform the terminological matching with the help of morphological and syntactic analysis to perform term normalisation. They are frequently used in Information Retrieval to improve searching. For example, they will find as similar classes `Match` and `Matching`. They operate on the principle of finding linguistic variants of the same string, as described above.

Morphological variants are most commonly identified through *stemming* algorithms, which strip words to their base form by removing suffixes such as plural forms and affixes denoting declension or conjugation. For example, `match`, `matching` and `matches` would all be reduced to the single stem `match`. The Porter stemming algorithm (or "Porter stemmer") [Porter, 1980] is often used for removing the more common morphological and inflectional endings from words in English. It has also been implemented for other languages such as German, and is freely available in many different formats and programming languages[1].

However, morphological conflation of terms is not just about suffix stripping. Morphological variations can also be expressed through different graphic transformations which have nothing to do with suffixes – or indeed any kind of affixes. This is referred to as *allomorphy*. Derived allomorphs, which are essentially a phonological mutation of the basic form, may still be recognisable using stemming techniques, e.g. "loaf" and "loaves". In the case of partial or total suppletion, however, the two allomorphs may bear only a partial resemblance to each other, or no resemblance at all (respectively), e.g. "bring – brought" or "go – went". In this case, a greedy clustering algorithm based on suffix strings may be used (for partial suppletion) [Jacquemin, 1997] or reference to external lists may be required.

Additionally, suffixes may be specific to a particular domain (e.g. medicine and biomedicine) and are therefore not considered by traditional stemmers such as [Porter, 1980] and [Lovins, 1968] or mentioned in the traditional literature on morphology, so domain-specific and/or ad-hoc methods may be required.

[1]http://www.tartarus.org

Jacquemin's method also deals with syntactic variants through the use of metarules, which state possible transformations that can be applied to particular classes of words. Syntactic transformations can only be applied to multi-word terms (at least, where one of the two terms in question is multi-word). For example, the rule

```
Metarule Perm (X1 --> X2 X3) = X1 --> X3 X4 X5 X2
```

enables the term "effect of glucose concentration" to be matched with the term "concentration effect", where X_1 can consist of either X_2 X_3 (glucose concentration) or X_3 X_4 X_5 X_2 (effect of glucose concentration).

This can be turned easily in a distance based on a cost model in which, for instance, term equality is 0, simple modifications are .5 and modifications are multiplied.

Extrinsic methods

Extrinsic methods make use of external resources such as dictionaries and lexicons. Lexicon-based methods essentially match terms which are semantically related, using an external lexicon or dictionary. For example, they consider synonyms as equivalent and hyponyms as subsumed, finding as similar classes `Match` and `Alignment`. Typically, WordNet is used – either to simply find close relationships such as synonymy between the two terms, or to compute some kind of semantic distance between them in order to decide if a relationship should hold. For example, [Patel *et al.*, 2003] use the principle of locality to discover relationships in WordNet between instances of concepts; [Su and Gulla, 2003] uses a semantic distance measure to strengthen the mappings of instances whose concept names are closely related in WordNet; [Silva and Rocha, 2003] also uses a semantic distance measure, adapted from that proposed by [Resnik, 1995], to compute similarities in order to establish correspondences between the terms, which are then used to transform instances from the source ontology into instances from the target ontology. A number of these methods have been implemented in a Perl package[2].

Simple measures can be defined here (we only consider synonyms because they are the basis of wordnet) but other relationships can be used. Moreover, the hyponym/hyperonym hierarchy is, in this respect, similar to a class hierarchy and the measure defined in § **??** can be used here. The simplest use of synonyms is in the following:

Definition 17 (Synonymy)**.** *Given two terms s and t and a synonym resource Σ, the synonymy is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$ such that:*

$$\sigma(s,t) = \begin{cases} 1 & \textit{if}\ \exists c \in \Sigma; s \in c \wedge t \in c \\ 0 & \textit{otherwise} \end{cases}$$

This strict exploitation of synonymy does not allow to discriminate when two objects are not synonyms, how far they are and when they are synonyms, how close they are. But, the synonymy being a relation, all the measures on the graph of relations can be used on wordnet synonyms, such as:

- compute the symmetric difference of the sets of synonyms of two terms;
- compute the size of the shortest path between two terms in the synonym graph;

[2]http://wn-similarity.sourceforge.net/

More elaborate measures take into account that the terms can be part of several "synset" and uses a measure in the "is-a" hierarchy between synsets. The two following ones are information-theoretic measures. Each synset (c) is associated a probability of occurence ($pi(c)$) of an instance of the concept associated to a particular synset. This probability is obtained from corpus study. It is obviously such that the more specific the concept, the lower its probability. The similarity between two terms is function of the more general synset common to both terms.

The similarity proposed in [Resnik, 1995; Resnik, 1999] maximises the information content or entropy (taken as the negation of the logarithm of the probability).

Definition 18 (Resnik semantic similarity)**.** *Given two terms s and t and a partially ordered synonym resource $\langle \Sigma, \leq \rangle$ provided with a probability π, Resnik semantic similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ such that:*

$$\sigma(s,t) = max_{k; \exists c,c' \in \Sigma; s \in c \wedge t \in c' \wedge c \leq k \wedge c' \leq k}(-log(\pi(k)))$$

Information-theoretic similarity [Lin, 1998] also implement a similarity which depends on the information content increase. This methods specifies the the probabilistic degree of overlap between two synsets:

Definition 19 (Information-theoretic similarity)**.** *Given two terms s and t and a partially ordered synonym resource $\langle \Sigma, \leq \rangle$ provided with a probability π, Dekang semantic similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ such that:*

$$\sigma(s,t) = max_{k; \exists c,c' \in \Sigma; s \in c \wedge t \in c' \wedge c \leq k \wedge c' \leq k} \frac{2 \times log(\pi(k))}{log(\pi(s)) + log(\pi(t))}$$

These similarities are not normalised.

Multilingual methods

Multilingual methods involve matching between terms in different languages, in order to create a multilingual ontology from two bilingual ones, or to align two multilingual ontologies. Typically they would make use of a multilingual dictionary such as EuroWordNet, though there are other possible methods. They use ideas and techniques from machine translation, clustering and monolingual ontology alignment.

There has been much work on semantic matching within a single language, but very little on cross-lingual semantic matching, i.e. measuring the semantic similiarity of words across languages. One of the main difficulties with this is that there can be many-to-many translations of words or terms, rather than a single direct correspondence. [Ngai *et al.*, 2002] proposes a method for multilingual ontology alignment based on the approach of [Fung and Lo, 1998] which uses word co-occurrence patterns to indicate semantic similarity. This method has the advantage of being able to use non-parallel bilingual corpora.

Ngai's approach is based on the assumption that even though each sense of a term may have different translations, i.e. there may be a one-to-many correspondence for translations of its different meanings, it is unlikely that its synonyms will have the exact same set of translations for each of their meanings. The approach considers the average similarity between terms in a synset (set of synonyns for that term) from one ontology and terns from all potential candidates for alignment from the other ontology. All candidate synsets are ranked according to a similarity measure, and the highest ranked set wins.

The Polylex project [3] aimed at creating a single multilingual lexicon for Dutch, English and German from individual monolingual lexicons (contained in the CELEX database[4]). Their approach relies on the fact that the 3 languages share many aspects of syntax, morphology, morphophonology, phonology and orthography. Their method makes use of orthogonal multiple inheritance, which allows a node in the hierarchy to inherit different kinds of information from different parent nodes. With this resource, each language's nodes can inherit a mix of information from within the language's own hierarchy and from the common hierarchy.

6.3 Structural (internal and external) methods

The structure of entities that can be found in ontology can be compared, instead of comparing their names or identifiers. This comparison can be subdivided in a comparison of the internal structure of an entity (i.e., its attributes or, for speaking OWL, the properties which takes their values in a data type) or the comparison of the entity with other entities to which it is related.

6.3.1 Internal structure

Methods based on the internal structure of entities use criteria such as the range of their properties (attributes and relations), their cardinality, and the transitivity and/or symmetry of their properties to calculate the similarity between them. Internal structure based methods are sometimes referred to as constraint based approaches in literature, e.g., [Rahm and Bernstein, 2001].

Entities with comparable internal structure or properties with similar domain and range in two ontologies can be numerous, that is why these kinds of methods are commonly used to create alignment clusters rather than to discover accurate correspondences between entities. They usually appear combined with other local methods like terminological, structural, or extensional ones and are in charge of reducing the number of align candidates. As stated by Li and Clifton in [Li and Clifton, 2000], methods comparing field specifications at the schema level do not intend to completely replace searching through a synonym lexicon, but help to determine attribute correspondences when no conclusion can be made simply by searching a synonym dictionary. They can be used with other approaches, as a 'first step' to eliminate most of the attributes that are clearly incompatible.

Compatibility do2002a

The Cupid algorithm for discovering mappings between schema elements [Madhavan *et al.*, 2001] depends, among others things, on the compatibility between datatypes of attributes which is assessed thanks to a lookup table. Identical data types have the highest compatibility value. Compatible type a compatibility value which does not disqualify them.

Data-based domain comparison

Data-based domain comparison is first an extensional method for aligning. It can be used for inductively finding domain information which is often missing in database schemas in which type structures are relatively poor.

[3]http://www.informatics.susx.ac.uk/research/nlp/polylex/polylex.html

[4]http://www.kun.nl/celex/index.html

SEMantic INTegrator (SEMINT) is a tool based on neural networks described in [Li and Clifton, 2000] to assist in identifying attribute correspondences in heterogeneous databases. SEMINT supports access to a variety of database systems and utilizes both schema information and data contents to produce rules for matching corresponding attributes automatically. The schema information used by SEMINT includes data types, length, scale, precision, and the existence of keys, value, and range constraints, disallowing null values, etc.

The instance data is used to compute some statistics like maximum, minimum, mean, variance, coefficient of variance, existence of null values, existence of decimals, scale, precision, grouping, and number of segments.

Other approaches to determine attribute correspondences using instance data try to compare attribute values. Larson et al. [Larson *et al.*, 1989] and Sheth et al. [Sheth *et al.*, 1988] discussed how relationships and entity sets can be integrated primarily based on their domain relationships: EQUAL, CONTAINS, OVERLAP, CONTAINED-IN, and DISJOINT. The problem is that determining such relationships can be time consuming and tedious. Another limitation is the ability to handle faults: small amounts of incorrect data may lead the system to draw a wrong conclusion on domain relationships. Other approaches like [Li and Clifton, 1994] proposes methods that utilize data patterns and distributions instead of data values and domains. The result is a better fault tolerance and less time-consumption since only a small portion of data values are needed by employing data sampling techniques. In general, applying internal structure methods to instances allow a more precise characterization of the actual contents of schema elements and thus, more accurately determine corresponding data types based, for example, on the discovered value ranges and character patterns.

Relative volume

Comparing the internal structure of objects amounts to compare their properties and to compose the obtained comparison index. The composition operation is considered in section 7.1. It can be used for composing the values of internal properties alone or to aggregate with the result of other similarities such as those resulting from the external structure.

Depending on the entities to be considered, the property values can be different: values in classes are domains while values in individuals are values. Moreover, these values can be structured in sets or sequences. It is thus important to consider this in the comparison.

[Valtchev, 1999] proposes a framework in which the types or domains of properties must be compared on the basis of their interpretations: sets of values. Type comparison is based on their respective size, in which the size of a type is the cardinal of the set of values it defines. The distance between two domains is then given by the difference between their size and that of their common generalization. This measure is usually normalized by the size largest possible distance attached to a particular datatype.

Definition 20 (Relative size distance)**.** *Given two domain expressions e and e' over a datatype τ, the relative size distance $\delta : 2^\tau \times 2^\tau \rightarrow [0,1]$, is such that:*

$$\delta(e,e') = \frac{|gen_\tau(e \vee e')| - 1/2 * (|gen_\tau(e)| + |gen_\tau(e')|)}{|\tau|}$$

in which $gen(.)$ provides the generalization of a type expression.

There are three advantages to this measure: the most obvious one is that it is normalized. The second one is that it is totally general. The third one is that can easily be mapped to the usual measures that are often used.

Usually, the common generalization depends on the type: it is a set for enumerated types, an interval for ordered types (it can also be a set of intervals). In case of dense types, the size of a domain is the usual measure of its size (Euclidean distance). The case of infinite types has to be taken adequately (by evaluating the largest possible domain in a computer or by normalizing with regard to the actual corpus). Normalizing over the actual largest distance in the corpus, if possible, is often a good idea. Indeed, it is not fair to normalise the age of people with that of planets or their size even if they use the same unit.

Another advantage of this framework is that is encompasses value comparisons which are compared as singletons.

Comparing multiplicities and properties

Another approach that compares attribute specifications using design information (the schema information) has been proposed in [Navathe and Buneman, 1986], the characteristics of attributes discussed are uniqueness, cardinality, domain, static semantic integrity constraints, dynamic semantic integrity constraints, security constraints, allowable operations, and scale.

Evaluating the compatibility of types on multiplicities is relatively easy: multiplicities are first interpreted as reducing the integer interval $[0 \ +\infty[$ and two multiplicities are compatible if the intersection of the corresponding intervals is non empty. Evaluating the similarity between the multiplicites of two properties can be achieved by the other methods presented here considering that they are interpreted as sets of integers.

In [Ehrig and Sure, 2004], Ehrig and Sure proposed the definition of a set of rules for determining similarity between ontology entities and point the fact that some features from OWL related to internal structure could be used, but are discarded by now, as they do not have any wide distribution yet. These features are property characteristics as symmetry and restrictions of values, among others.

Similarity between collections

It is often necessary to compare sets or lists of objects (e.g., the set of children of someone or the sequence of meals in a menu). In this case, general techniques can be used for assessing the similarity or distance between these sets depending on the similarity applying to the type of their elements. Concerning sets, these methods will be presented in section 6.4 in the context of extension comparison. Concerning sequences, they can be adapted from some of the measures that have been presented in section 6.2.1 which have considered strings as sequences of characters and paths as sequences of strings. In addition, some of the methods of § 6.4 can also be applied to sequences.

These observations apply both to internal and external structure.

6.3.2 External structure

The similarity comparison between two entities from two ontologies can be based on the position of entities within their hierarchies. If two entities from two ontologies are similar, their neighbours might also be somehow similar. This remark can be used in several different ways. Criteria for deciding that the two entities are similar include:

C1 Their direct super-entities (or all of their super-entities) are already similar [Dieng and Hug, 1998a].

C2 Their sibling-entities (or all of their sibling-entities) are already similar.

C3 Their direct sub-entities (or all of their sub-entities) are already similar [Dieng and Hug, 1998a].

C4 All (or most) of their descendant-entities (entities in the subtree rooted at the entity in question) are already similar.

C5 All (or most) of their leaf-entities (entities, which have no sub-entity, in the subtree rooted at the entity in question) are already similar [Madhavan *et al.*, 2001].

C6 All (or most) of entities in the paths from the root to the entities in question are already similar [Bach *et al.*, 2004].

Of course, an approach can combine several of the above criteria [Mädche and Staab, 2002; Bach *et al.*, 2004].

External structure comparison faces problems when the viewpoint of two ontologies is highly different (see Deliverable D2.1.1). For example, with the same class “Human”, in the first ontology, it can be specialized into two sub-classes “Man” and “Woman” but in the second ontology, it can be divided into “Adult” and “Young_Person”. In this case, the application of this method is not a good solution.

The methods for aggregating the external structure features of entities are very similar to those to be used in case of internal structure. However, one can find some particularities when dealing with partially ordered domains and, in particular, with hierarchies. Methods specifically related to hierarchical domains are considered here.

Mereologic structure

In a mereologic hierarchy, the relations between entities are whole-part relations. The sub-entity is a “part” of the super-entity, and vice versa, the super-entity can be composed of some different sub-entities. For example, a super-class “Desktop_Computer” can have some whole-part relations with a sub-class “Motherboard”, with a sub-class “Graphics_Card”, with a sub-class “CPU”... The application of criterion [C5] for computing the similarity between entities from different ontologies with mereologic structure does not seem to be convenient here. The other criteria may still be applied.

Taxonomic structure

In a taxonomic hierarchy, the relations between entities are specialisation relations. The sub-entity is a “specialisation” of the super-entity, and vice versa, the super-entity is a “generalisation” of the sub-entity. A super-entity can have relations with one or more sub-entities and similarly, a sub-entity can have relations with one or more super-entities. For example, a super-class “Motor” can have some specialisation classes such as “Motocycle”, “Passenger Vehicle”. As an other example, class “Researcher” is a generalisation class of two sub-classes “Research Fellow” and “Senior Researcher”.

Contrary to mereologic structure, the similarity computation between entities from different ontologies with taxonomic structure can apply criterion [C5]. The application of above criteria

[C1-6] can tell us that the class "Researcher" may be similar to the class "Research Staff Member" if the latter has also two classes "Research Fellow" and "Senior Researcher" as its specialisation.

There have been several measures proposed for comparing classes based on the taxonomic structure. The *structural topological dissimilarity* δ^s on a domain [Valtchev and Euzenat, 1997] follows the graph distance, i.e. the shortest path distance in a graph (taken here as the transitive reduction of the hierarchy).

Definition 21 (structural topological dissimilarity on hierarchies)**.** *The structural topological dissimilarity* $\delta : O \times O \rightarrow \mathbb{R}$ *is a dissimilarity over a hierarchy* $H = \langle O, \leq \rangle$*, such that:*

$$\forall e, e' \in O, \delta(e, e') = \min_{c \in O}[\delta(e, c) + \delta(e', c)] \tag{6.1}$$

where $\delta(e, c)$ *is the number of intermediate edges between an element* e *and another element* c*.*

This corresponds to the unit tree distance of [Barthélemy and Guénoche, 1992] (i.e., with weight 1 on each edge). The corresponding normalized function is:

$$\overline{\delta}(e, e') = \frac{\delta(e, e')}{\max_{o, o' \in O} \delta(o, o')} \tag{6.2}$$

The result given by such a measure is not always semantically relevant since a long path in a class hierarchy can often be summarized as a short one.

The upward cotopy distance had been described in [Mädche and Zacharias, 2002] as follows.

Definition 22 (upward cotopic distance)**.** *The upward cotopic distance* $\delta : O \times O \rightarrow \mathbb{R}$ *is a dissimilarity over a hierarchy* $H = \langle O, \leq \rangle$*, such that:*

$$\delta(c, c') = \frac{|UC(c, H) \cap UC(c', H)|}{|UC(c, H) \cup UC(c', H)|} \tag{6.3}$$

where $UC(c, H) = \{c' \in H; c \leq c'\}$ *is the set of superclasses of* c*.*

Of course, these measures cannot be applied as such in the context of ontology alignment since the ontologies are not supposed to share the same taxonomy H (but this can be used in conjunction with a common resource such as wordnet). For that purpose, it is necessary to develop these kinds of measure over a pair of ontologies. In [Valtchev, 1999; Euzenat and Valtchev, 2004], this amounts to use a (local) matching between the elements to be compared (for instance, the hierarchies).

Relations

The similarity computation between entities can be also based on their relations. If class A relates to class B by relation R in one ontology, and if class A' relates to class B' by relation R' in the other ontology, and if we know that B and B' are similar, R and R' are similar, we can infer that A and A' may be similar too. By the same way, if A is similar to A', R is similar to R', B may be similar with B'; or R may be similar with R' if we know before that A and A' are similar, B and B' are similar: the similarity among relations in [Mädche and Staab, 2002] is computed according to this principle. For example, classes "Company" and "University" will be considered similar because they have a similar relation "hasEmployee" with class "Employee" and class "Professor" which are themselves similar.

This can be extended to a set of classes and a set of relations. It means that if we have a set of relations $R_1 \dots R_n$ in the first ontology which are similar with an other set of relations $R'_1 \dots R'_n$ in the second ontology, it is possible that two classes, which are the domains of relations in those two sets, are similar too.

One of the problems for this approach is to define how two relations are similar. This approach is based on the similarity of relations to infer the similarity of their domain classes or their range classes. Relations between classes in an ontology can be considered as entities in that ontology, they can be organized in a relation hierarchy, and like classes, the similarity computation between relations is also a big problem.

Remark: Both above problems in which we compare the similarity of entities in mereologic hierarchy or taxonomic hierarchy can be considered as a sub-case of the last problem where the relations between entities (classes) are only whole-part relations or is_a/specialisation relations.

6.4 Extensional (based on instances)

Extension-based methods compares the extension of classes, i.e., their set of instances rather than their interpretation. There are two very different conditions in which such techniques can be used: when the classes share the same instances (§ 6.4.1) and when they do not (§ 6.4.2).

6.4.1 Common extension comparison

The easiest way to compare classes A and B when they share classes is to test their intersection and to consider that these classes are very similar when $A \cap B = A = B$, more general when $A \cap B = B$ or $A \cap B = A$. However, the dissimilarity can only be 1 when none of these cases apply, for instance if the classes have some instance in common but not all. A way to refine this is to use of the Hamming distance between the two extension: it corresponds to the size of the symmetric difference normalized by the size of the union.

Definition 23 (Hamming distance)**.** *The Hamming distance between two sets is a disimilarity function* $\delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$, $\delta(x, y) = \frac{|x \cup y - x \cap y|}{|x \cup y|}$.

This version of the symmetric difference is normalized.

It is also possible to compute a similarity based on the probabilistic interpretation of the set of instances. This is the case of the Jaccard similarity.

Definition 24 (Jaccard similarity)**.** *Given two sets A and B, let* $P(X)$ *the probability of a random instance to be in set* X*, the Jaccard similarity is defined by:*

$$\sigma(A, B) = \frac{P(A \cap B)}{P(A \cup B)}$$

This measure is normalized and reaches 0 when $A \cap B = \emptyset$ and 1 when $A = B$.

6.4.2 Similarity-based extension comparison

Similarity-based techniques do not ask the classes to share the same set of instances (however, they can still be applied in that case). In particular, the above methods always return 0 when the two classes do not share any instances, disregarding the distance between the elements of the sets.

In some case, it is preferable to assess the distance between these classes. In order to compare the set of instances they use a (dis)similarity between the instances which can be computed with any of the methods presented here.

In data analysis, the linkage aggregation methods allows to assess the distance between two sets whose objects are only similar.

Definition 25 (Single linkage)**.** *The single linkage measure between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,* $\Delta(x,y) = \min_{(e,e') \in x \times y} \delta(e,e')$*.*

Definition 26 (Full linkage)**.** *The complete linkage measure between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,* $\Delta(x,y) = \max_{(e,e') \in x \times y} \delta(e,e')$*.*

Definition 27 (Average linkage)**.** *The average linkage measure between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,* $\Delta(x,y) = \frac{\sum_{(e,e') \in x \times y} \delta(e,e')}{|x| * |y|}$*.*

Each of these methods have its own benefits. Another methods from the same familly is the Hausdorff distance measuring the maximal distance of a set to the nearest point in the other set:

Definition 28 (Haussdorf distance)**.** *The Haussdorf distance between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,*

$$\Delta(x,y) = max(\max_{e \in x} \min_{e' \in y} \delta(e,e'), \max_{e' \in y} \min_{e \in x} \delta(e,e'))$$

6.4.3 Matching-based comparison

The problem with the former distances, but average, is that their value is function of the distance between one couple of members of the set. The average linkage on the opposite has its value function of the distance between all the possible comparison.

Matching-based comparisons [Valtchev, 1999] consider that the element to be compared are those which are corresponds to each others, i.e., the most similar one.

To that extent, the distance between two sets is considered as a value to be minimized and its computation as an optimization problem: the one of finding the elements of both sets which corresponds to each others. This corresponds to solving the square assignment problem.

Definition 29 (Match-based similarity)**.** *The match-based similarity between two sets is a similarity function* $MSim : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,*

$$MSim(x,y) = \frac{\sum_{\langle n,n' \rangle \in Pairing(x,y)} \sigma(n,n')}{max(|x|,|y|)}$$

in which $Pairing(x,y)$ *is a mapping of elements of* x *to elements of* y *which maximises the group* $MSim$ *similarity.*

This match-based similarity already require an alignment of entities to be computed. It also depends on the kind of mapping that is required. Indeed, the result will be different if the mapping is required to be injective or not.

The match-based comparison can also be used when comparing sequences. See [Valtchev, 1999] for a complete discussion on that topic.

6.5 Semantic methods (based on models)

The key characteristics of semantic methods is that they have model-theoretic semantics which is used to justify their results. Hence they are deductive methods. Examples are propositional satisfiability (SAT) and modal SAT techniques or description logic based techniques.

As from [Giunchiglia and Shvaiko, 2003b; Giunchiglia and Yatskevich, 2004; Bouquet and Serafini, 2003] the approach of applying propositional satisfiability (SAT) techniques to alignment is to translate the matching problem, namely the two tree-like structures (e.g., concept hierarchies) and mapping queries into a propositional formula and then to check it for its validity. By mapping query we mean here the pair of nodes and a possible relation between them. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability, and therefore will exhaustively check for all possible mappings.

Modal SAT can be used, as proposed in [Shvaiko, 2004b], for extending the methods related to propositional SAT to binary predicates. Its basis is to delimit propositional SAT from the case of trees which allows handling only unary predicates (e.g., classes) by admitting binary predicates (e.g., slots, etc.). The key idea is to enhance propositional logics with modal logic (or a kind of description logics) operators. Therefore, the matching problem is translated into a modal logic formula which is further checked for its validity using sound and complete satisfiability search procedures.

Description logics techniques, i.e. subsumption test, can be used to establish the relations between classes in a purely semantic manner. In fact, first merging two ontologies (after renaming) and then testing each pair of concepts and role for subsumption is enough for aligning terms with the same interpretation (or with a subset of the interpretations of the others).

Of course, pure semantic methods do not perform very well alone, they often need a preprocessing phase providing “anchors”, i.e., entities which are declared to be equivalent (based on their name or human input for instance).

These methods being semantically exact do only provide a similarity of 1 for objects considered equivalent. However, they allow for more variety in the expression of the correspondence between entities such as establishing that one entity satisfies all the models of another or that two entities cannot share any instance.

Chapter 7

Global methods

Once the local methods for determining the similarity or (dis)similarity are available, there remain to compute the alignment. This involve some kind of more global treatments, including:

- aggregating the results of these base methods in order to compute the similarity between compound entities (§ 7.1);
- developing a strategy for computing these similarities in spite of cycles and non linearity in the constraints governing similarities (§ 7.2);
- organising the combination of various similarity/alignment algorithms (§ 7.4).
- involving the user in the loop (§ 7.5);
- finally extracting the alignments from the resulting (dis)similarity: indeed, different alignments with different characteristics can be extracted from the same (dis)similarity (§ 7.6).

Moreover, we will consider more global techniques of learning how to align ontologies from example (§ 7.3).

All these steps are considered here under the name of global methods. They combine local methods in order to define an original algorithm for providing an alignment.

7.1 Compound similarity

Compound similarity is concerned with the aggregation of local (and compound) similarities. As a matter of fact, some objects are understood as compound and their (dis)similarity depends on that holding between their components (the similarity between two classes may depend on the similarity of their names, their super-classes and their properties).

7.1.1 Classical distances and weighted sums

In case the difference between some properties must be aggregated, one of the more common falilly of distances are the Minkowski distances

Definition 30 (Minkowski distance)**.** *Let O a set of objects which can be analized in n dimensions, the Minkowski distance between two such objects is:*

$$\forall x, x' \in O, \delta(x, x') = (\sum_{i=1}^{n} \delta(x_i, x'_i)^p)^(1/p)$$

in which $\delta(x_i, x'_i)$ is the dissimilarity of the pair of objects along the i^{th} dimension.

Instances of the Minkowski distances are the Euclidean distance (when $p = 2$), the City-block (a.k.a. Manhattan) distance (when $p = 1$) and the Chebichev distance (when $p = +\infty$).

These distances can be weighted in order to give more importance to some parameters. They can be normalized by dividing their results by the maximum possible distance (which is not always possible).

It has the main drawback of not being linear if $p \neq 1$, see [Valtchev, 1999] for a discussion of the consequences.

The simple linear aggregation can be further refined by adding weights to this sum. Weighted linear aggregation does consider that some of the values to be aggregated do not have the same importance (for instance, similarity in properties is more important than similarity in comments). The aggregation function will thus use a set of weights $w_1, \ldots w_n$ corresponding to a category of entities or properties. The aggregation function is then:

Definition 31 (Weighted sum)**.** *Let O a set of objects which can be analized in n dimensions, the weighted sum (or weighted average) between two such objects is:*

$$\forall x, x' \in O, \delta(x, x') = (\sum_{i=1}^{n} w_i * \delta(x_i, x'_i)$$

in which $\delta(x_i, x'_i)$ is the dissimilarity of the pair of objects along the i^{th} dimension and w_i is the weight of dimension i.

In fact, the weights can be different depending on the categories of the object aggregated as well as that of the similarity computed [Euzenat and Valtchev, 2004]. Then, the function can use a set of weights w_C^P depending of the category of object C and the kind of value computed P.

This kind of measure can be normalized, if all values are normalized, by having:

$$\sum_{i=1}^{n} w_i = 1$$

.

7.1.2 Triangular norms

Triangular norms are used as conjunction operators in uncertain calculi.

Definition 32 (Triangular norm)**.** *A triangular norm T is a function from $D \times D \rightarrow D$ (with D a set ordered by $\leq$ and provided with an upper bound $\top$) satifying:*

$$\begin{aligned} T(x, \top) &= x && (boundary\ condition) \\ x \leq y \implies T(x, z) &\leq T(y, z) && (monotonicity) \\ T(x, y) &= T(y, x) && (commutativity) \\ T(x, T(y, z)) &= T(T(x, y), z) && (associativity) \end{aligned}$$

There are typical examples of the triangular norms: $min(x, y)$, $x.y$ and $max(x + y - 1, 0)$. All are normalized if the measures provided to them are normalized; min is the only idempotent norm ($\forall x, min(x,x) = x$) and the values are ordered by $min(x,y) \geq x.y \geq max(x+y-1, 0)$. Moreover, any triangular norm can be expressed as a combination of the these three functions [Hájek, 1998].

The triangular norms can be extended to n-ary measures. On instance of such a generalization using both weights and n arguments is the weighted product.

Definition 33 (Weighted product)**.** *Let O a set of objects which can be analized in n dimensions, the weighted product between two such objects is:*

$$\forall x, x' \in O, \delta(x,x') = \prod_{i=1}^{n} \delta(x_i, x'_i)^{\lambda_i}$$

in which $\delta(x_i, x'_i)$ is the dissimilarity of the pair of objects along the i^{th} dimension and λ_i is the weight of dimension i.

These operators have the drawback that if only one of the dimension has a measure of 0, then the result is 0.

7.1.3 Weighted averages (and fuzzy aggregates)

The weighted average is very often used as a fuzzy aggregate [Gal *et al.*, 2004]. We will see why below.

Definition 34 (Fuzzy aggregate operator)**.** *A fuzzy aggregate operator f is a function from $D^n \rightarrow D$ (with D a set ordered by $\leq$ and provided with an upper bound $\top$) satifying:*

$$\begin{aligned} f(x, \ldots x) &= x && (idempotency) \\ \forall x_i, y_i \text{ such that } x_i \leq y_i, f(x_1, \ldots x_n) &\leq f(y_1, \ldots y_n) && (increasingmonotonicity) \\ f \text{ is a continuous function} & && (continuity) \end{aligned}$$

A typical example of a fuzzy aggregate operator is the weighted average.

Definition 35 (Weighted average)**.** *Given a set $w_1, \ldots w_n$ of weigths,*

$$\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

is a weighted average function.

The simple average function is such a function with all weigths equals. Again, if the values are normalized, the weighted average is normalized.

These kinds of function are very useful if ones want to use a learning algorithm for learning the weights of the measure.

The use of neural networks (NNs) in order to realize the key concepts of a fuzzy logic system enriches the system with the ability of learning and improves the subsymbolic to symbolic mapping. Neural network realization of basic operations of fuzzy logic, such as fuzzy complement, fuzzy intersection and fuzzy union, has been proposed ([Pao, 1989])[Hsu et al., 1992]. The activation function of neurons is then set to be one of the three basic operations mentioned above in order to provide fuzzy logic inference. Another approach is to use the ordered weighted averaging neuron [Yager, 1992] for representing fuzzy aggregating operations. A feedforward network can also be used to represent the membership function of a fuzzy set.

Fuzzy logic inference systems can be used to represent complex relations of subsymbolic to symbolic mapping by defining possibility distributions on the antecedents and the consequents of if-then rules. The nonadaptive and heuristic behavior of fuzzy logic systems can be improved with the aid of NNs. For this aim, a connectionist approach of fuzzy inference has been proposed in [Keller and Hunt, 1992]. The network is referred to as *fuzzy inference network* and implements a rule of the form.

$$x_1 \in A_1 \wedge x_2 \in A_2 \wedge \dots A_n \in a_n \implies y \in B$$

For choosing the parameters of the fuzzy inference that are associated with the parameters of the network, a training algorithm has been proposed.

7.2 Global similarity computation

The computation of compound similarity is still local because it only provides similarity considering the neighbourhood of a node. However, similarity may involve the ontologies as a whole and the final similarity values may ultimately depend on all the ontologies. Moreover, the distance defined by local methods can be defined in a circular way (for instance if the distance between two classes depends on the distances between their instances which themselves depends on the distance between their classes or if there are circles in the ontology). In case of circular dependencies, similarity computation is not anymore possible in a local fashion. It is more like an optimization problem.

For that purposes, strategies must be defined in order to compute this global similarity. The first one is defined as a process of propagating the similarity within a graph while the second one translate the similarity definitions in a set of equations which is solved by classical techniques.

7.2.1 Similarity flooding

Similarity flooding [Melnik *et al.*, 2002] is a generic graph matching algorithm which uses fix-point computation to determine corresponding nodes in the graphs.

The two ontologies are first translated into directed labelled graphs grounding on the OIM specification [MDC, 1999]. The principle of the algorithm is that the similarity between two nodes must depend on the similarity between their adjacent nodes (whatever are the relations that must be taken into account). To implement this, the algorithm creates another graph G whose nodes are pairs of nodes of the initial graphs and there is an edge between (o_1, o'_1) and (o_2, o'_2) labeled by p whenever there are edges (o_1, p, o_2) in the first graph and (o'_1, p, o'_2) in the second one. So, the alignment does only take into account edges with the same label.

Then, the algorithm computes initial similarity values between nodes (based on their labels for instance) and then iterates steps of re-computing the similarities between nodes in function of the similarity between their adjacent nodes at the previous step. It stops when no similarity changes more than a particular threshold ϵ or after a predetermined number of steps.

The chosen aggregation function is a weighted linear aggregation in which the weight of an edge is the inverse of the number of other edges with the same label reaching the same couple of entities.

$$\sigma^{i+1}(x,x') = \frac{\sum_{((x,x'),p,(y,y'))\in G}\sigma^i(y,y')}{|\{(y,y')|((x,x'),p,(y,y'))\in G\}|} + \frac{\sum_{((y,y'),p,(x,x'))\in G}\sigma^i(y,y')}{|\{(y,y')|((y,y'),p,(x,x'))\in G\}|}$$

The values are further normalised with regard to the maximal similarity value obtained (i.e., that value is assigned the value 1. and the other values are reduced proportionally).

7.2.2 Similarity equation fixpoint

In many situations, e.g., with symmetric or inverse properties, it is impossible to establish an ordering of entities in order to compute the similarities in a step-wise manner. [Euzenat and Valtchev, 2004] provides a method for dealing with circularities and dependencies between similarity definition which is described hereafter. In this case, the similarity values can only be expressed as a set of equations where each variable corresponds to the similarity between a pair of nodes. There is an equal number of equations, each of them associated to a variable. The structure of each equation follows the definition of the respective similarity function for the underlying node category.

Some facts are worth mentioning. First, there is no need for a different expression of the similarity functions in case there are no effective circular dependences between similarity values. In fact, the computation mechanism presented below establishes the correct similarity values even if there is an appropriate ordering of the variables (the ordering is implicitly followed by the step-wise mechanism). Moreover, in case some similarity values (or some similarity or (dis)similarity assertions) are available beforehand, the corresponding equation can be replaced by the assertion/value.

If each of the similarity expression is a linear aggregation of other similarity variables, this system would be solvable directly because all variables are of degree one.

However, in the case of OWL-Lite, and of many other languages, the system is not linear since there could be many candidate pairs for the best match. The similarity may depend on matching the multiple edges with the similar labels outgoing from the nodes under consideration. In this approach, the similarity is computed by an $MSim$ function that first finds an alignment between the set of considered entities and then computes the aggregated similarity in function of this matching.

Given two classes c and c', the resulting class similarity function reads as follows:

$$\begin{aligned}\sigma_C(c,c') &= \pi^C_L\sigma_L(\lambda(c),\lambda(c')) \\ &+ \pi^C_O MSim_O(\mathcal{I}(c),\mathcal{I}'(c')) \\ &+ \pi^C_S MSim_C(\mathcal{S}(c),\mathcal{S}'(c')) \\ &+ \pi^C_P MSim_P(\mathcal{A}(c),\mathcal{A}'(c'))\end{aligned}$$

The function is normalised since weights are, i.e., $\pi^C_L + \pi^C_S + \pi^C_O + \pi^C_P = 1$, whereas each factor that ranges over collections of nodes/feature values is averaged by the size of the larger collection.

Nevertheless, the resolution of the resulting system can still be carried out as an iterative process that simulates the computation of the greatest fixed point of a vector function, as shown by Bisson [Bisson, 1992]. The trick consists in defining an approximation of the $MSim$-measures, solving the system, replacing the approximations by the newly computed solutions and iterating.

The first values for these $MSim$-measures are the maximum similarity found for a pair, without considering the dependent part of the equations. The subsequent values are those of the complete similarity formula filled by the solutions of the system. Note that the local matching may change from one step to another depending of the current similarity values.

However, the system is converging because the similarities can only grow (because the non dependant part of the equation will remain and all dependencies are positive) and, in case of similarity values are bounded (e.g., to 1) the similarity is bounded. The iterations will stop when no gain above a particular ϵ value is provided by the last iteration. If the algorithm converges, we cannot guarantee that it does not stop in a local optimum (that is finding another matching in the $MSim$-measures would not increase the similarity values). This could be improved by randomly changing these matchings when the algorithm stops.

This methods has some similarity with the previous one: both methods work iteratively on a set of equations extracted from a graphical form of the ontologies. Both methods ultimately depends on the computed proximities between non-described language elements, i.e., data type names, values, URIRefs, property type names, etc. Indeed, these proximities are propagated throughout the graph structure by the similarity dependancies.

However, Similarity flooding is dependent on the edge labels, while the latter method takes similarity between properties into account. Nonetheless it also considers local mappings between alternative matching edges instead of averaging over all the potential match. Moreover, the Similarity flooding method is stated so generally that its convergence is not proved.

7.3 Learning methods

Like in many other fields, learning methods developed in machine learning reveals useful in ontology alignment. In order to achieve a global similarity that can be used for aligning, techniques created for machine learning can be used instead of the one presented above. They are mainly used in two particular areas:

- supervised learning in which the ontology alignment algorithm learns how to work through the presentation of many good alignment (positive examples) and bad alignments (negative examples). Of course, the alignment can be provided by the algorithm itself. This approach is not really used so far, the reason being that it is difficult to know which techniques works well for which ontology features, so an ontology alignment algorithm learnt with several ontology pairs, might not necessarily work well for a new ontology pair.
- learning from data in which a population of instances is communicated to the algorithm together with theirs relations and the classes they belong to. From this data, the algorithm can learn the relations between classes (specialisation) and the alignment of properties.

Both techniques usually take advantage of well-known methods in machine learning: formal-concept analysis [Stumme and Mädche, 2001], Bayes learning [Berlin and Motro, 2002] or neural networks [Li and Clifton, 1994].

7.3.1 Learning from probabilistic distribution

The Glue system [Doan *et al.*, 2004; Doan, 2002] is based on learning classifiers for classes from instances in order to evaluate the joint probability distributions of instances.

The principle is that two classes are more likely to be the same if their instances are the same. So it goal is to evaluate the probability for a random instance to be a member of both classes ($P(c, c')$). More precisely, for each couple of classes c and c', the system requires the computation of $P(c, c')$, $P(c, \overline{c'})$, $P(\overline{c}, c')$ and $P(\overline{c}, \overline{c'})$ (in which $\overline{c}$ is the complement of c, i.e., reads "not in c"). This is useful because some similarities, e.g., the Jaccard similarity (see § 6.4) can be rewritten as:

$$\frac{P(c, c')}{P(c, c') + P(c, \overline{c'}) + P(\overline{c}, c')}$$

Joint probability distributions could be estimated if both ontologies shared the same set of instances. However, when this is not the case, the algorithm learns a classifier for each class in each ontology and uses the result for attributing classes to the instances of the other ontology. It can then estimate the joint probability distributions.

In Glue, there are several learners, which are trained by data instances of ontologies. They use different criteria to evaluate the reason to belong to a class. After learning phase, different characteristic instance patterns and matching rules for single elements of the target schema are discovered. The predictions of individual matchers are combined by a meta-learner, and from that, assignment of classes to instances will be deduced.

From the probability distribution the algorithm can use a probabilistic metrics for assessing similarities between classes.

7.3.2 Learning (fuzzy) aggregation through neural networks

Another approach is the generalization of the Sugeno-Takagi inference model [1988]. *Neural-network driven fuzzy reasoning*, proposed by Takagi and Hayashi [Hayashi *et al.*, 1992], constructs the antecedent part of a fuzzy system using a back-propagation network. A lot of interesting ideas, useful in symbolic to subsymbolic mapping, can be found in this approach and especially in the steps of selection of input-output variables and clustering of the training data.

The issue of identifying the fuzzy rules and tuning the membership functions of fuzzy sets using neural networks and training algorithms has been widely studied. Horikawa et al. [Horikawa *et al.*, 1992] proposed *fuzzy modeling networks* in order to realize the fuzzy reasoning process through association of its parameters with the weights of a backpropagation learning neural network. According to this method, the basic parameters of the fuzzy model can be automatically identified by modifying the connection weights of the network. Another approach to this problem related to fuzzy control has been proposed by Lin and Lee [Lin and Lu, 1995] (chapter 19), who introduced the *fuzzy adaptive learning control network* (FALCON) to study hybrid structure-parameter learning strategies. Structure learning algorithms are used to find appropriate fuzzy rules and parameter learning algorithms are used to fine tune the membership functions and other parameters of the fuzzy inference system. Actually, Lin and Lee proposed a number of different architectures and learning procedures. The first model, the FALCON-H, is a multi-layer feedforward network which represents in a connectionist structure the basic elements and functions of a fuzzy logic controller. It is supported by a hybrid learning algorithm that consists of two separate stages, combining unsupervised learning and supervised gradient-descent learning (backpropagation). Structure learning, which can lead to the extraction of rules, is based on the adaptive fuzzy partitioning of the input and output spaces and the combination (association) of these partitions. In order to provide a more flexible fuzzy partitioning, Lin and Lee proposed the FALCON-ART model, applying adaptive resonance theory (ART) learning. The above models require precise training data to indicate the desired output through a supervised learning process. Unfortunately, such type of data may be very difficult or even impossible to obtain (especially for the feature-to-symbol-mapping). The FALCON-R model, developed to remedy the above problem, is a reinforcement learning neurofuzzy system that performs automatic construction based on a right-wrong (reward-penalty) binary signal. The applied learning algorithm of FALCON-R is complicated, since it is designed to use deterministic reinforcement feedback, stochastic reinforcement feedback and reinforcement feedback with long time delay. Berenji and Khedkar [Berenji and Khedkar, 1992] have proposed another neurofuzzy control system using reinforcement learning, the *generalized approximate reasoning-based intelligent controller* (GARIC). All the above models have been well tested for fuzzy control applications. Their advantage is that they have the ability to support automatically the extraction of fuzzy inference rules. Multi-layer perceptrons have also been used for multistage fuzzy inference based on the propagation of linguistic truth values [Uehara and Fujise, 1992].

7.3.3 Semantic Gossiping

Semantic gossiping [Aberer *et al.*, 2003b] is an approach to establish global semantic agreements in any emergent way and is based on (1) local mappings among the schemas of the participating parties and their propagation and (2) analysis of the quality of these mappings. To simplify presentation we will assume a peer-to-peer (P2P) system in the following.

We assume that groups of peers have already agreed on common semantics, i.e., a common schema. We denote these groups as *semantic neighborhoods*. If two peers located in two disjoint neighborhoods meet, e.g., during query processing and forwarding, they can exchange their schemas and provide translations between them. We assume that the translations are provided by the users or through any feasible approach for alignment. During the life-time of the system, each peer has the possibility to learn about existing translations and add new ones. This means that incrementally a directed graph of translations will be built among the peer schemas along with the normal operation of the system.

This translation graph has two interesting properties: (1) based on the already existing translations and the ability to learn about translations, queries can be propagated to peers for which no direct translation link exists by means of transitivity and (2) the graph will have cycles. We call (1) *semantic gossiping*. (2) gives us the possibility to assess the degree of *semantic agreement* along a cycle, i.e., to measure the quality of the translations and the degree of semantic agreement in a community.

We expect peers to perform several task: (1) upon receiving a query, a peer has to decide where to forward the query to, based on a set of criteria overviewed below; (2) upon receiving results or feedback along translation cycles, it has to analyze the quality of the results at the schema and at the data level and adjust its criteria accordingly; and (3) update its view of the overall semantic agreement by modifying its query forwarding criteria or by adjusting the translation themselves.

The criteria to assess the quality of translations—which in turn is a measure of the degree of semantic agreement—can be categorized as *context-independent* and *context-dependent*. Context-independent criteria are syntactic in nature and relate only to the transformed query and to the required translation. We use the notion of *syntactic similarity* to analyze the extent to which a query is preserved after transformation. Context-dependent criteria relate to the degree of agreement that can be achieved among different peers upon specific translations. Such degrees of agreement may be computed using feedback mechanisms. We use two such feedback mechanisms, namely cycles appearing in the translation graph and results returned by different peers. This means that a peer will locally obtain both returned queries and data through multiple feedback cycles. In case a disagreement is detected (e.g., a wrong attribute mapping at the schema level or a concept mismatch at the content level), the peer has to suspect that at least some of the translations involved in the cycle were incorrect, including the translation it has used itself to propagate the query. Even if an agreement is detected, it is not clear whether this is not accidentally the result of compensating mapping errors along a cycle. Thus, analyses are required that assess which are the most probable sources of errors along cycles, to what extent the own translation can be trusted and therefore of how to use these translations in future routing decisions. At a global level, we can view the problem as follows: The translations between domains of semantic homogeneity (same schemas) form a directed graph. Within that directed graph we find cycles. Each cycle allows to return a query to its originator which in turn can make the analysis described above.

Each of these criteria is applied to the transformed queries and evaluated and results in a *feature vector* (see [Aberer *et al.*, 2003b] for details). The decision whether or not to forward a query using a translation link, i.e., whether a translation is assumed to be correct or not, is then based on evaluating these feature vectors.

Based on this approach the network converges to a state where a query is only forwarded to the peers most-likely understanding it, where the correct translations are increasingly reinforced by adapting the per-hop forwarding behaviors of the peers and where incorrect translations are rectified. Implicitly, this is a state where a global agreement on the semantics of the different schemas has been reached. Experimental results [Aberer *et al.*, 2003b] indicate that semantic agreement can be reached in a network of partially erroneous translations.

7.4 Method composition

Similarity values provided by local methods have to be aggregated. However, alignment and similarity assessment methods are also aggregated in order to compose a particular algorithm. For instance, one can first compute similarities between class names, then compute similarity between properties depending on how their names and classes to which they are attached are similar and then run a fix-point algorithm for computing interdependent similarities.

We can distinguish three ways to compose these methods:

built-in composition corresponds to most of the algorithms presented in next chapter: the chaining of methods is part of the algorithm and is applied to any data set which is given to the system.

opportunistic composition would correspond to a system which chooses the next method to run in function of the input data. We are not aware of any system working in that manner.

user-driven composition is used in environments in which the user has many different methods that she can apply following her will.

There are not a lot to be said on these three ways to compose the available methods. One system that proposes some starting point is Rondo [Melnik *et al.*, 2003b]. Its goal is to propose a languages for composing the schemas and morphism. It thus defines Match and Merge operations (among many operations on their schemas ? Union, Difference ? and morphisms ? Compose, Invert). However, these two operators are placeholders for various matching and merging strategy which can be implemented as a combination of these other operators and SQL queries.

Most of the individual methods can be composed through a general purposes programming language.

7.5 User input

The support of effective interaction of the user with the system components is one concern of ontology alignment. User input can take place in many areas of alignment:

– for assessing initial similarity between some terms;
– for invoking and composing alignment methods (see above);
– for accepting or refusing similarity or alignment provided by the various methods.

7.5.1 Relevance feedback

In particular, the user feedback for each specific mapping extracted by the system could provide the alignment system with the ability of improving itself by changing the local alignment system parameters of the aggregation of the local alignments. However, asking users to specify this information is in general difficult since people conceptions change through their interaction with the alignment system. In any case, the main aspects of the user interaction could be summarised into the following questions:

– Given an alignment provided by the system, what is the structure and the way that the system can represent the user judgment for this specific alignment?
– How can the system take advantage of the above user feedback in terms of improving itself with the aid of this information?

Thus, the steps in implementing the user feedback are a) to gain an understanding of the nature of the this feedback and b) to specify an alignment system design and structure that supports and enhance it.

7.6 Alignment extraction

The ultimate alignment goal is a satisfactory set of correspondences between ontologies. A (dis)similarity measure between the entities of both ontologies provides a first set of correspondences. Those which will be part of the resulting alignment remains to be extracted with the help of the computed similarity.

This can be achieved during any of the global methods above if sufficiently constrained (for instance, to retain only one correspondence per entity). This can also be achieved afterwards by a specialised extracting method.

An alignment can be obtained by displaying the entity pairs with their similarity scores and/or ranks and leaving the choice of the appropriate pairs up to the user of the alignment tool. This user input can be taken as the definitive answer in helper environments, as the definition of an anchor for helping the system or as relevance feedback in learning algorithms.

One could go a step further and attempt at defining algorithms that automate alignment extraction from similarity scores. Various strategies may be applied to the task depending on the properties of the target alignment. As a matter of fact, one can ask the alignment to be complete (total) for one of the ontologies, i.e., all the entities of that ontology must be successfully mapped on the other side. Completeness is purposeful whenever thoroughly transcribing knowledge from one ontology to another is the goal. One can also require the mapping to be injective and hence reversible.

7.6.1 Thresholds

If neither ontology needs to be completely covered by the alignment, a threshold-based filtering would allows us to retain only the most similar entity pairs. Without the injectivity constraint, the pairs scoring above the threshold represent a sensible alignment.

The easier way to proceed consists in selecting correspondences over a particular threshold. Several methods can be found in the litterature [Ehrig and Sure, 2004]:

Hard threshold retains all the correspondence above threshold n;

Delta method consists in using as a threshold the highest similarity value to which a particular constant value d is substracted;

Proportional method consists in using as a threshold the a percentage of the highest similarity value;

Percentage retains the $n\%$ correspondences above the others.

7.6.2 Optimizing the result

In contrast, if an injective mapping is required then some choices need to be made in order to maximize the "quality" of the alignment that is typically measured on the total similarity of the aligned entity pairs. Consequently, the alignment algorithm must optimize the global criteria rather than maximizing the local similarity at each entity pair.

To sum up, the alignment computation may be seen as a less constrained version of the basic set similarity functions $MSim$ (see § 6.4.3). Indeed, its target features are the same: (i) maximal total similarity, (ii) exclusivity and (iii) maximal cardinality (in entity pairs). However, (ii) and (iii) are not mandatory, they depend on injectivity and completeness requirements, respectively.

A greedy alignment algorithm could construct the correspondences step-wise, at each step selecting the most similar pair and deleting its members from the table. The algorithm will then stop whenever no pair remains whose similarity is above the threshold.

The greedy strategy is not optimal: finding the global optimum would require the computing of a square assignment (polynomial assignment algorithms are suggested in [Papadimitriou and Steiglitz, 1998]). However, the ground on which a high similarity is forgotten to the advantage of lower similarities can be questioned and thus the greedy algorithm could be preferred in some situations.

Chapter 8

System presentations

The various methods presented above in isolation have been put together in order to implement ontology alignment or schema matching systems. There are a number of available systems that can be seen as addressing ontology alignment. We present some of them below through their principles and availability. Some of the following systems are developed by the projects partners and thus will be usable in order to benchmark them in the future.

There were some comparisons of these systems, in particular in [Do *et al.*, 2002; Rahm and Bernstein, 2001; Kalfoglou and Schorlemmer, 2003b; Parent and Spaccapietra, 2000]. Our purpose here is not really to compare them, but rather to show their variety. Table 8.1 summarizes the kind of techniques implemented in each of these systems.

8.1 Prompt and Anchor-Prompt (Stanford SMI)

The Anchor-PROMPT [Noy and Musen, 2001] (an extension of PROMPT, also formerly known as SMART) is an ontology merging and alignment tool with a sophisticated prompt mechanism for possible matching terms. The anchor-PROMPT alignment algorithm takes as input two ontologies and a set of anchors-pairs of related terms, which are identified with the help of string-based techniques, or defined by a user, or another matcher computing linguistic (dis)similarity between frame names (labels at nodes), for example [McGuinness *et al.*, 2000]. Then it refines them based on the ontology structures and users feedback.

It constructs a directed labeled graph representing the ontology from the hierarchy of concepts (called classes in the algorithm) and the hierarchy of relations (called slots in the algorithm), where nodes in the graph are concepts and arcs are relations denoting relationships between concepts (the labels are the names of the relations). An initial list of anchors-pairs of related concepts defined by the users or automatically identified by lexical matching is the input for the algorithm. Anchor-PROMPT analyzes then the paths in the sub-graph limited by the anchors and it determines which concepts frequently appear in similar positions on similar paths. Based on these frequencies, the algorithm decides if these concepts are semantically similar concepts.

The PROMPT and Anchor-PROMPT systems have also contributed to the design of other algorithms such as PROMPTDiff which finds differences between two ontologies and provides the editing operation for transforming one ontology into another.

Page	System	T	TS	TL	I	S	ST	SC	E	M	U
65	**Multikat**	x		x	x	x	x				
61	**FCA-Merge**						x		x		
62	**IF-map**						x		x		
58	**APrompt**	x			x	x	x				x
60	**Cupid**	x	x	x	x	x	x				
73	**QOM**	x	x		x	x	x	x	x		
68	**OLA**	x	x		x	x	x	x	x		
60	**Rondo**		x			x		x			x
62	**T-tree**					x			x		
63	**S-match**	x	x	x			x			x	
65	**Buster**				x		x			x	
61	**Glue**								x		
60	**Chimerae**	x	x	x						x	x
62	**Artemis**	x		x		x					
329	**Coma**	x	x			x					x
68	**Asco**	x	x	x			x				
60	**MoA**			x							x
69	**Dogma**	x	x	x							
71	**ArtGen**			x							
72	**Bibster**	x	x		x		x				
75	**KILT**									x	

Table 8.1: Various contributions to alignment at a glance. The columns corresponds to categories of Chapter 6: Terminological (string- or language-based); Internal structure; Structure (terminological and cyclic); Extensional; semantics (Model-based) and User.

8.2 Chimerae (Stanford KSL)

Chimaera is an environment for merging and testing (diagnosing) large ontologies [McGuinness *et al.*, 2000]. Matching in the system is performed as one of the major subtasks of a merge operator. Chimaera searches for merging candidates as pairs of matching terms, involving term names, term definitions, possible acronym and expanded forms, names that appear as suffixes of other names. It also has techniques to identify terms that should be related by subsumption, disjointness, etc.

8.3 Rondo (Stanford U./U. Leipzig)

Rondo [Melnik *et al.*, 2003b] is an environment for model (e.g., database schema) engineering which provides many unit primitives for manipulating models (extract, restrict, delete) and way to compose them. Among the unit primitives is the implementation of Similarity flooding (see § 7.2.1). It converts schemas (SQL DDL, XML) into directed labeled graphs whose nodes are candidate aligned pairs and arcs are shared properties. Arcs are weighted by their relevance to the nodes.

8.4 MoA (ETRI)

MOA[1] is an environment for merging ontologies developped by Electronics and Telecomunication Research Institute (ETRI) in South Korea. It is a library of methods and a shell for using them. It can work on OWL (but does not tell which flavor) and contains methods for importing, aligning, modifying and merging ontologies. Unfortunately, the methods are not known beside that they are based on (dis)similarity. The system uses Jena and Wordnet.

8.5 Cupid (Microsoft research)

The Cupid system [Madhavan *et al.*, 2001] implements a generic schema matching algorithm combining linguistic and structural schema matching techniques, and computes normalized similarity coefficients with the assistance of a precompiled thesaurus. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. Matching algorithm consists of three phases and operates only with tree-structures to which no-tree cases are reduced. The first phase (linguistic matching) computes linguistic similarity coefficients between schema element names (labels) based on morphological normalization, categorization, string-based techniques and a thesaurus look-up. The second phase (structural matching) computes structural similarity coefficients which measure the similarity between contexts in which individual schema elements occur in the schemas under consideration. The main idea behind the structural matching algorithm is to rely more on leaf level matches instead of the immediate descendents or intermediate substructures when computing similarity between non-leaf elements. The third phase (mapping generation) computes weighted similarity coefficients and generates final mappings by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold. In comparison with the other hybrid matchers e.g., Dike [Palopoli *et*

[1]http://mknows.etri.re.kr/moa

al., 2000] and Artemis (see 8.9), referring to [Madhavan *et al.*, 2001], Cupid performs better in the sense of mapping quality.

8.6 Glue (U. of Washington)

Glue [Doan, 2002] is an evolved version of LSD [Doan *et al.*, 2001] whose goal is to semi-automatically find schema mappings for data integration. Like its ancestor LSD, Glue use machine learning techniques to find mappings [Doan *et al.*, 2004]. It first applies statistical analysis to the available data (joint probability distribution computation). Then generates a similarity matrix, based on the probability distributions, for the data considered and use "constraint relaxation" in order to obtain an alignment from the similarity (see § 7.3.1). The algorithm works in three steps:

learning distributions the first phase is described above(see § 7.3.1), it learns the joint probability distributions of classes of each ontologies;

similarity estimation the system estimates the similarity between two classes in function of their joint probability distributions.

relaxation produces an alignment from the similarity matrix by using heuristic rules for choosing the more likely correspondences.

8.7 FCA-merge (U. Karlsruhe)

FCA-merge [Stumme and Mädche, 2001] uses formal concept analysis techniques to merge two ontologies sharing the same set of instances. The overall process of merging two ontologies consists of three steps:

1. instance extraction,
2. concept lattice computation, and
3. interactive generation of the final merged ontology.

The algorithms theoretically merges two ontologies sharing the same set of instances. However, the authors provide, as first step, methods for extracting the instances from documents. The extraction of instances from text documents circumvents the problem that in most applications there are no individuals which are simultaneously instances of the source ontologies, and which could be used as a basis for identifying similar concepts.

The computation of the lattice starts with two ontologies and instances belonging to both ontologies. From these, it computes two formal contexts, i.e., boolean tables indicating which instance belongs to which concept of the of the ontology. It then merges both contexts (by renaming the concepts and adding both contexts). Using classical formal concept analysis (i.e., the closure of an instances$\times$properties Galois connection [Ganter and Wille, 1999]) on contexts made of instances$\times$concepts, the method generates a pruned concept lattice. The lattice is pruned of all the concepts which are not more general than a concept of one of the ontologies.

The last step consists in helping a user to further simplify the lattice and generate the taxonomy of an ontology. The produced result is explored and transformed to a merged ontology by the ontology engineer. The final step of deriving the merged ontology from the concept lattice requires human interaction.

The result is rather a merge than an alignment. However, the concepts that are merged can be considered as exactly aligned and those which are not can be considered in subsumption relation with their ancestors or sibblings.

8.8 IF-Map

Another system inspired by formal concept analysis is IF-Map [Kalfoglou and Schorlemmer, 2003a]. It is an automatic method for ontology mapping based on the Barwise-Seligman theory of information flow [Barwise and Seligman, 1997]. The basic principle of IF-map is to align two local ontologies by looking at how these are mapped from a common reference ontology. It is assumed that such reference ontology is not populated with instances, while local ontologies usually are. IF-Map generates possible mappings between an unpopulated reference ontology and a populated local ontology by taking into account how local communities classify instances with respect to their local ontologies.

8.9 Artemis (U. Milano/U.Modena and Reggio Emilia)

Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [Castano *et al.*, 2000] was designed as a module of MOMIS mediator system [Bergamaschi *et al.*, 1999; Bergamaschi *et al.*, 1998] for creating global views. Artemis does not cover all the issues of matching due to the origin function of schema integration. The matching algorithm performs affinity-based analysis and hierarchical clustering of source schemas elements. Affinity-based analysis is carried out through computation of the name, structural and global affinity coefficients by exploiting a common thesaurus. The common thesaurus presents a set of terminological and extensional relationships which depicts intra- and inter-schema knowledge about classes and attributes of the input schemas, which is built with the help of WordNet [Miller, 1995] and ODB-Tools [Beneventano *et al.*, 1998]. A hierarchical clustering technique exploiting global affinity coefficients categorizes classes into groups at different levels of affinity. For each cluster it creates a set of global attribute global class. Logical correspondence between the attributes of a global class and source attributes is determined through a mapping table.

8.10 T-tree (INRIA Rhne-Alpes)

Troeps [Mariño *et al.*, 1990] was a knowledge representation system enabling several class taxonomies (called viewpoints) over the same set of objects and bridges between these classes expressed equivalence or subsumption. T-tree [Euzenat, 1994] is an environment for generating taxonomies and classes from objects (instances). It can, in particular, infer dependencies between classes (bridges) of different ontologies sharing the same set of instances based only on the "extension" of classes.

An algorithm has been developed which is able to infer bridges. The bridge inference algorithm, given a set of source viewpoints and a destination viewpoint (built by T-Tree or by any other mean), returns all the bridges (in a minimal fashion) which are satisfied by the available data. That is the set of bridges for which the objects in every source class are indeed in the destination class. The algorithm compares the extension (set of instances) of the presumed destination to the intersection of these of the presumed source classes. If there is no inclusion of the latter in the former, the algorithm is re-iterated on all the sets of source classes which contain at least one class which is a sub-class of the tested source classes. If the intersection of the extension of the presumed source classes is included in that of the presumed destination class, then a bridge can be established from the latter (and also from any set of sub-classes of the source classes) to the former (and also any super-class of the destination class). But other bridges can exists on the sub-classes of the destination. The algorithm is thus re-iterated on them. It stops when the bridge is trivial, i.e. when the source is empty.

The algorithm is extension-correct (only valid bridges are inferred), extension-complete (all valid bridges are inferred) and extension-minimal (only more general bridges are inferred). The proof is carried out in the classification scheme framework and the "extension-" prefix just tells that what is considered is only the extension of the classes (the algorithm tests set inclusion on classes). Thus these results are not semantically grounded. For instance, is that a coincidence that all directors have formerly been at the same university? Maybe, maybe not. Hence the user has to decide the validation of inferred bridges. This has to be contrasted with a stronger kind of bridge inference based on the structural constraints on classes. But indeed, any possible bridge compliant with the current set of objects and the semantics must be a restriction of one of the bridges provided by the algorithm.

Bridge inference is nothing else than the search for correlation between two sets of variables. This correlation is particular from a data analysis point of view since it does not need to be valid on the whole set of individuals (the algorithm looks for subsets under which the correlation is valid) and it is based on strict set equality (not similarity). However, even if the bridge inference algorithm has been described with set inclusion, it can be helped by other measurements which will narrow or broaden the search. More generally, the inclusion and emptiness tests can be replaced out by tests based on the similarity of two sets of objects (as it is usual in data analysis). In fact, many parameters can be taken into account when inferring bridges; for that purpose, the algorithm is function of the meaning of the operators $\subseteq$, $\cap$ and $= \emptyset$-test. A second version of the algorithm (with the same properties) were made available and used structural comparison: $\subseteq$ is subtyping, $\cap$ is type intersection and $= \emptyset$-test is a sub-typing test.

8.11 S-MATCH (U. Trento)

S-Match is a schema/ontology matching system that implements the semantic matching approach [Giunchiglia and Yatskevich, 2004; Bouquet and Serafini, 2003]. It takes two graph-like structures (e.g., database schemas or ontologies) as input and returns semantic relations between the nodes of the graphs, that correspond semantically to each other, as output. Possible semantic relations are: equivalence ($=$), more general ($\sqsupseteq$), less general ($\sqsubseteq$), mismatch ($\perp$) and overlapping ($\sqcap$).

The current version of S-Match is a rationalized re-implementation of the CTXmatch system [Bouquet and Serafini, 2003] with a few added functionalities. S-Match is schema based, and, as such, it does not exploit the information encoded in data instances. S-Match is a hybrid system performing composition of element level techniques. At present, S-Match allows it to handle only tree-like structures (e.g., taxonomies or concept hierarchies).

S-Match was designed and developed as a platform for semantic matching, namely a highly modular system with the core of computing semantic relations where single components can be plugged, unplugged or suitably customized. The logical architecture of the system is depicted in Figure 8.1.

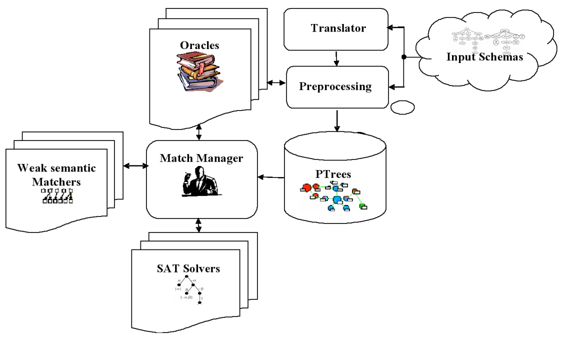


Figure 8.1: Architecture of the S-match platform

The *input schemas* (trees) are codified in a standard internal XML format. This internal format can be loaded from a file that is manually edited, or can be produced by an input format dependent translator. The module taking input schemas/ontologies does the preprocessing. In particular, it computes in a top-down manner for every label in a tree the meaning captured by the given label in a schema or ontology using the techniques described in [Magnini *et al.*, 2003]. The preprocessing module has access to the set of oracles which provide the necessary a priori lexical and domain knowledge. In the current version WordNet [Miller, 1995] is the only oracle. The output of the module is an enriched tree. These enriched trees are stored in an internal database (PTrees) where they can be browsed, edited and manipulated.

The Matching Manager coordinates matching process using three extensible libraries. The first library is contained of, what is called in [Giunchiglia and Yatskevich, 2004], weak semantics element level matchers. They perform string manipulations (e.g., prefix, n-grams analysis, edit distance, soundex, data types, and so on) and try to guess the semantic relation implicitly encoded in similar words. The current version of S-Match contains 13 weak semantics element level matchers. The second library is contained of strong semantics element level matchers, namely oracles. Currently, WordNet is the only oracle. The third library is contained of structure level strong semantics matchers, namely SAT solvers (among the others, the SAT deciders that we are currently testing is JSAT [Le Berre, 2001] and Open4J by Daniel Le Berre).

S-Match is implemented in Java 1.4 and the total amount of code (without optimizations!) is around 60K.

8.12 Coma (U. Leipzig)

The COMA system [Do and Rahm, 2002] is a generic schema matching tool, which implements composite generic matchers. COMA provides an extensible library of matching algorithms; a framework for combining obtained results, and a platform for the do2002a of the effectiveness of the different matchers. Matching library is extensible, and as from [Do and Rahm, 2002] it contains 6 individual matchers, 5 hybrid matchers, and one “reuse-oriented” matcher. Most of them implement string-based techniques as a background idea; others share techniques with Cupid (see § 8.5) but reuse-oriented is a completely novel matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as rooted directed acyclic graphs, where elements are the paths. This aims at capturing contexts in which the elements occur. One of the distinct features of the COMA tool is the capability to perform iterations in matching process. It presumes interaction with a user which approves obtained matches and mismatches to gradually refine and improve the accuracy of match.

Based on the comparative do2002as conducted in [Do *et al.*, 2002], COMA dominates Autoplex&Automatch [Berlin and Motro, 2001; Berlin and Motro, 2002] LSD [Doan *et al.*, 2001], Glue [Doan *et al.*, 2003b], SF [Melnik *et al.*, 2002] and SemInt [Li and Clifton, 1994] matching tools.

8.13 Buster (U. Bremen)

The Bremen University Semantic Translator for Enhanced Retrieval (BUSTER) [Visser *et al.*, 2002] is an information broker middleware that was built to access heterogeneous and distributed information sources and to assess their conceptual, spatial, and temporal relevance with respect to a specific information request. BUSTER can also be used to integrate heterogeneous information through the resolution of structural, syntactical, and semantic heterogeneities. To be more precise, the BUSTER system provides two subsystems, one for information filtering and one for information integration.

The BUSTER search module supports the specification of queries of the type concept @ location in time [Vögele *et al.*, 2003]. In addition to the conceptual semantics, the system evaluates the spatial as well as the temporal relevance of an information source. In order to be able to reason about conceptual, spatial, and temporal relevance, BUSTER utilises metadata that provide formal descriptions of the respective context of an information source.

In principle, the main difference with respect to other system for query processing and information integration lies in the fact that the user does commit to a basic vocabulary that is used to define concepts in all the source ontologies. The basic vocabulary ensures that different source ontologies are comparable to each other. By formulating the query in terms of this basic vocabulary we the query can be interpreted with respect to all source ontologies in the system. In particular, each concept base on the shared vocabulary and can be constructed with the help of some construction operators like $\sqcap, \sqcup$ well-known from description logics. Because each concepts also from different source ontologies can be flatten to terms which only consists of elements of the shared vocabulary combined with some construction operators, they can easily compared with respect to equality ($\equiv$), subsumption ($\sqsubseteq$), overlap ($C \sqcap D$ is consistent), and inconsistence ($C \sqcap D$ is inconsistent). In other words, BUSTER can automatically determine these concepts in a source ontology that are most similar to the concept we asked for.

8.14 MULTIKAT (INRIA Sophia Antipolis)

MULTIKAT [Dieng and Hug, 1998a; Dieng and Hug, 1998b] is a tool enabling comparison and merging of two ontologies, represented in Sowa's conceptual graph formalism [Sowa, 1984]. In this formalism, an ontology is represented through a support (i.e. a hierarchy of concept types, a hierarchy of relation types, a set of markers for identifying the instances and a conformity relation enabling to determine which types are compatible with a given marker).

The building of an integrated ontology from two ontologies relies on the following steps:

1. Comparison and merging of the two concept type hierarchies: this step enables to solve

name conflicts and in case of need, to add new concept types and to adapt concept type definitions.

2. Comparison and merging of the two relation hierarchies: this step enables to solve name conflicts, and in case of need, to add new relation types, to adapt relation type definitions and to adapt relation type signatures.
3. Comparison and merging of the two sets of markers: this phase helps to solve name conflicts and to adapt the conformity relation.

MULTIKAT relies on a cooperative approach: the knowledge engineer can use MULTIKAT editor to tune the parameters and weights used in the mapping and merging algorithms.

8.14.1 Mapping of two types in both hierarchies

The mapping algorithm aims at determining, in two concept (resp. relation) type hierarchies, which types are identical. It relies on two phases:

Phase 1: terminology-based mapping During this first phase, MULTIKAT algorithm tries to identify which types of both hierarchies are similar, according to their main names and their synonyms. The knowledge engineer can combine several criteria and assign them different weights so as to privilege some criteria:

- t_1 and t_2 have the same main name,
- the number of common synonyms of t_1 and t_2 is greater than a given threshold,
- the main name of one type belongs to the list of synonyms of the other type.

This similarity function $Sim_1 : H_1 \times H_2 \rightarrow \mathbb{R}$ computes the similarity measure $Sim_1(t_1, t_2)$ between t_1, a type of H_1 and t_2, a type of H_2, according to this first identification phase, and its results are stored in a similarity matrix. After this phase, two types t_1 and t_2 are 1-similar iff Sim1 (t1, t2) is greater than a threshold Tsimilar.

Phase 2: context-based mapping In this second phase, the mapping algorithm now considers the contexts of the types to be compared. The context of a type consists of its relatives (i.e. its direct supertypes and its direct subtypes) in the type hierarchy. In this second phase, the algorithm tries to identify which types of both hierarchies are the same, according to their contexts. Three mapping cases are distinguished:

- The number of 1-similar direct supertypes (resp. direct subtypes) of t_1 and t_2 are greater than a threshold $Tpred$ (resp. $Tsucc$)
- All the direct supertypes (resp. direct subtypes) of t_1 and t_2 are 1-similar.
- The set of relatives of t_1 (resp. t_2) is included in the set of relatives of t_2 (resp. t_1) w.r.t. 1-similarity.

The knowledge engineer can associate different weights to these three cases. Another similarity function $Sim_2(t_1, t_2)$ is computed. If t_1 is the type numbered i in Hier1 and t_2 the type numbered j in Hier2, then, in the final similarity matrix $SimMatr$:

$$SimMatr_{ij} = Sim_1(t_1, t_2) + Sim_2(t_1, t_2)$$

The couples of identical types are computed from this similarity matrix. After the second phase, the types t_1 and t_2 are considered as identical iff $SimMatr_{ij}$ is the maximum value in the ith line and the jth column in the matrix, and this value is greater than a threshold Tsame.

Two comparison strategies can be applied:

One-to-one algorithm For each cycle of comparison of two previous identification phases, the algorithm compares each type of H_1 to each type of H_2.

Hierarchy-match algorithm This algorithm takes into account the hierarchical structure in its comparison strategy. It relies on a depth-first search in both hierarchies and it proceeds as follows: once two identical types have been found, then a search for further mappings in their sub-hierarchies is performed. In this algorithm, the thresholds $T_{similar}$ and T_{same} have the same values. In both previous phases, after each do2002a of a couple of types (t_1, t_2), the corresponding value $SimMatr_{ij}$ is compared to T_{same}. As soon as $SimMatr_{ij} > T_{same}$, then the pair (t_1, t_2) is included in the set IdenticalConceptTypes.

8.14.2 Merging of concept type hierarchies

The knowledge engineer can initialize the set IdenticalConceptTypes by indicating which types of both hierarchies are already known as identical. The mapping algorithm is applied (either with a one-to-one match strategy or with a hierarchy match strategy). Then, before the merging, the partial ordering relation of identical types is checked. The couples, responsible for violation of the merging precondition are eliminated from IdenticalConceptTypes. Then the integrated hierarchy T_{ccom} is built by representing each couple of identical types by a single type in T_{ccom} and by adding the types appearing in only one hierarchy. If a type is present only in one ontology and cannot be mapped to any type of the second ontology, it will be kept in the integrated hierarchy, with a prefix in its name indicating from which expert it comes from. If a type is present in both hierarchies, the experts can choose its final main name stored in T_{ccom}. In all cases, the associated synonyms are also stored in T_{ccom}.

The algorithm tries to detect terminology conflicts, topology conflicts and conflicts specific to conceptual graph formalism.

8.14.3 Comparison and merging of the relation type hierarchies

The mapping algorithm for relation type hierarchies is similar to the 2-phase-based algorithm previously presented. Once obtained the set IdenticalRelationTypes of pairs of identical relation types, the precondition for merging of the two hierarchies must also be checked. When two relation types are considered as identical, a verification of their signature compatibility must be performed. The signatures in the integrated relation type hierarchy Trcom must be adapted according to the integrated concept type hierarchy Tcom.

If a relation type is present only in one ontology, its signature is preserved in the integrated ontology. The signature of the integrated relation type obtained from two identical relations types relies on the supremum of the concept types appearing in their signatures.

8.14.4 Comparison and merging of the marker sets

The terminology-based mapping algorithm is used for the set of markers. When two markers are identical, their conformity relation must be compatible, otherwise they are eliminated from the set IdenticalMarkers.

8.14.5 Implementation

MULTIKAT was implemented in C/C++ and JAVA, above the conceptual graph platform, COGITO (developed by the LIRMM) and was applied in traffic accident analysis.

8.15 ASCO (INRIA Sophia-Antipolis)

ASCO prototype relies on an algorithm that identifies the pairs of corresponding elements in two different ontologies [Bach *et al.*, 2004]. These pairs may be pairs of concepts (classes) in the two ontologies or pairs of relations, or even pairs of a concept in one ontology and a relation in the other ontology.

ASCO tries to use as much as possible available information contained in the ontology for the process of matching two ontologies. This information consists of identifiers (names), labels, comments of concepts, identifiers, labels, comments, domain, range of relations, structure of the taxonomy of concepts or of relations, data instances of ontology, annotations, axioms, rules. So far, in its matching process, ASCO already takes into account some of above information such as identifiers, labels, comments of concepts, identifiers, labels, comments, domain, range of relations, structure of the taxonomy of concepts or of relations.

The matching process of ASCO is composed of several phases. The linguistic phase applies linguistic processing techniques, and uses string comparison metrics, and lexical databases such as WordNet to compute the similarity of two concepts or two relations. In the linguistic processing step, ASCO normalizes firstly terms, expressions thanks to punctuation, upper case, special symbols, digits to have a set of tokens. These tokens are then compared using string comparison metrics such as Jaro-Winkler, Levenstein or Monger-Elkan. Based on token similarities, the similarity of sets of tokens is computed. To increase the accuracy and to avoid the problems of term conflicts, a lexical database such as WordNet is integrated. To compute the similarity between long texts (for example, between the comments or descriptions of classes or of relations), ASCO uses Term frequency/Inverse document frequency metrics after applying a linguistic processing step to eliminate all of the stopwords in long texts.

The computed linguistic similarities are input for the structural phase. In this phase, ASCO tries to exploit the structure of ontology taxonomy for modifying or asserting the similarity of two concepts or relations. The similarities of classes or of relations are iteratively propagated to their neighbors in the tree of ontology which is built from the hierarchy of classes and the hierarchy of relations. When the propagation terminates (the class similarities and the relation similarities do not change after an iteration or a certain number of iterations is reached), if the similarities between classes or relations exceed a threshold, they are considered as similar. ASCO runs now on the two above phases.

ASCO algorithm was implemented in Java. It is built on Corese (Conceptual Resource Search Engine), the semantic search engine developed by ACACIA team [Corby *et al.*, 2000; Corby and Faron, 2002; Corby *et al.*, 2004]. Corese loads ontologies from RDF(S) files into memory, these ontologies are then supplied to ASCO. ASCO was tested with two real-world ontologies: O'COMMA, which has 472 concepts and 77 relations [Gandon, 2002]; and O'Aprobatiom, which has 460 concepts and 92 relations.

8.16 OLA (INRIA Rhne-Alpes & UoMontréal)

OLA [Euzenat and Valtchev, 2003; Euzenat and Valtchev, 2004] is a class of algorithm for ontology alignments which targets the following characteristics:

- covering all the possible characteristics of ontologies (i.e., terminological, structural and extensional);
- taking care of collection structures (lists, sets) and accounting for them during matching;
- expliciting all recursive relationships and finding the best matching through iteration.

OLA is currently implemented for ontologies described in OWL-Lite [Euzenat and Valtchev, 2003]. It uses the Alignment API and implementation that we recently developed [Euzenat, 2004].

The algorithm first compiles the OWL ontologies into graph structures unveiling all relationships between entities. These graph structures produce the constraints for expressing a similarity between the elements of the ontologies. The similarity between nodes of the graphs follows two principles: (i) it depends on the category of node considered (e.g., class, property), and (ii) it takes into account all the features of this category (e.g., superclasses, properties). This similarity is a weighted linear aggregation of the similarity measures between all the entities a couple of entities is in relation. This accounts for all the relationships between entities. However, these features (like subclasses) are sets of entities, the similarity between these sets of entities, thus depends on a local matching between these entities. A matching of both sets is considered which is: (i) of maximal total similarity, (ii) exclusive, and (iii) of maximal size [Valtchev, 1999].

Similarity between labels can be produced by any kind of particular terminological method (e.g., string distance, linguistic do2002a). Similarity between data values and data types can be provided by specilised external similarity measures (e.g., Euclidean distance, symmetric difference distance) [Valtchev, 1999].

The definition of this similarity provides a set of equations whose variables are the similarity values between entities of the ontologies. This set of equation cannot be solved directly due to local matching. As a matter of fact, depending on the currently computed similarity, the matching as defined above can be different. We thus developed an iterative algorithm which compute a first approximation of the similarity (without the local matching), then compute the local matching and reiterate. We proved that this algorithm is converging towards a solution, mainly because the similarity is always improving over the iterations. It can be that this solution is not the global optimum so the algorithm should be launched several times.

From this solution, it is possible to extract an alignment between the two ontologies (by retaining the correspondence whose similarity is over a certain threshold, or by optimising the selections of couples).

8.17 Dogma's Ontology Integration Methodology

Dogma's ontology integration methodology adopts for ontology integration the same methodological steps that were singled out in database schema integration [Batini *et al.*, 1986]:

$$\begin{array}{ccccccc} \text{Step 1} & & \text{Step2} & & \text{Step3} & & \text{Step4} \\ \text{Preintegration} & \longrightarrow & \text{Alignment} & \longleftrightarrow & \text{Conforming} & \longrightarrow & \text{Merging and Restructuring} \end{array}$$

The double sided arrow between step 2 and 3 denotes a loop. An alignment rule is suggested in step 2 and step 3 checks if it would cause inconsistencies when approved and stored in the alignment rule repository.

Preintegration

Preintegration consists of an analysis of the ontologies to decide the general integration policy: choosing the ontologies to be integrated, choosing the strategy of integration, deciding the order of integration, and possibly assigning preferences to entire ontologies or portions thereof. All these decisions have to be made by humans. We adopt the *binary ladder strategy* to integrate ontologies [Batini *et al.*, 1986].

Comparison of Ontologies: Ontology Alignment

An *alignment* between two Dogma inspired ontologies is defined as a commitment (i.e. an interpretation) of the source ontology's lexon base in terms of the target ontology's lexon base. A commitment consists of a set of *commitment rules*, here (for ontology integration purposes) also called *mapping rules*. We discuss these mapping rules in more detail in the sections that follow.

The fundamental activity in this step consists of detecting and resolving several kinds of heterogeneities like semantically equivalent concepts which are denoted by means of different terms in both ontologies and identifying inter ontology relationships between concepts of different ontologies.

Identifying equivalent concepts: Because ontology mismatches often occur through ambiguity of terms, miss-spelled terms, usage of different terminology to denote the same concept etc., we will always consider concepts in the comparison phase instead of the term labels that represent them. The degree of similarity between two concepts c_1 and c_2 is measured by means of a *similarity score*, formally stated as: $\mathbf{sc(c_1, c_2) : C \times C \rightarrow [0, 1]}$. The way this similarity score is computed depends on how the concepts are defined. This is discussed next.

In case the concepts are WordNet synsets we can make use of the freely available software package, called *WordNet::Similarity*[2][Pedersen *et al.*, 2004], to measure the semantic similarity and relatedness between a pair of concepts.

If the concepts cannot be identified with existing WordNet synsets we compute the similarity score between two concepts $c_1 \equiv t^1_1, \ldots, t^1_n$ and $c_2 \equiv t^2_1, \ldots, t^2_m$ by computing the similarity score between all possible pairs of terms where the two terms in a pair come from different concepts. A similarity score between natural language terms is the weighted sum of similarity scores based on natural language and string processing techniques, like: Porter Stemmer, Metaphone, Levenshtein, longest common prefix/suffix, longest common substring, the theory of distributionalism. All these techniques are based on syntactic differences between terms and do not take any semantic value of them into consideration. Therefore we have to be very critical with the interpretation of these results. If the similarity score is above a given threshold then the concepts are considered to be equivalent. The treshold can be modified by the expert performing the alignment.

[2]http://sourceforge.net/projects/wn-similarity

Identifying inter ontology relationships: In order to identify inter ontology relationships between concepts of different ontologies we distinguish the following set of relationships with predefined semantics: *SubClassOf, Generalize, PartOf, InstanceOf*. We have developed a methodology that allows to automate the task of finding inter ontology relationships between concepts [De Bo *et al.*, 2004a; De Bo *et al.*, 2004b].. This methodology uses a formal upper level ontology, called SUMO (Suggested Upper Merged Ontology), which has been proposed as the initial version of an eventual Standard Upper Ontology (SUO)[Niles and Pease, 2001]. In a nutshell the methodology works as follows: Each concept used in the ontology is aligned with an appropriate SUMO concept. Since SUMO is a formal upper level ontology we can make use of its axioms to derive relations that hold between SUMO concepts to the ontology level.

Conforming of Ontologies: Instant Validation

Each time an alignment rule is proposed by the system or the user the rule is instantly checked whether it conflicts with other rules that have already been added to the alignment rule repository. In order to resolve conflicts in an automatic manner algorithms have been developed that detect conflicting situations caused by misalignments, look for cycles, etc.

Ontology Merging and Restructuring

During this activity the (conformed) ontologies are superimposed, thus obtaining a global ontology. The merge process is essentially based upon the mapping rules established in the comparison phase 8.17 and results in an algebra for merging. We distinguish following merge operators: *Merge, Specialize, Generalize, PartOf, InstanceOf*. All operators might require restructuring of the merged ontology.

8.18 ArtGen (Stanford U.)

In [Mitra and Wiederhold, 2002] the authors propose a semi-automated algorithm for resolving terminological heterogeneity among the ontologies and establishing the articulation rules necessary for meaningful interoperation. This algorithm forms the basis of the *articulation generator* for the ONION (ONtology compositION) system. The automated articulation generator (ArtGen) of ONION suggests articulation rules to the user performing the matching process. A human expert can either accept, modify or alter the suggestions. The expert can also indicate new matches that the articulation generator might have missed.

The authors distinguish two types of articulation rules:

linguistic matching rules Concept names are represented as a string of words. The linguistic matcher compaires all possible pairs of words from any two concepts of both ontologies and assigns a similarity score to each pair. The matcher uses a word similarity table generated by a word relator (Thesaurus-Based word relator or Corpus-Based word relator) to look up the similarity between all possible pairs of words. The similarity score between two concepts is the average of the similarity scores (different from zero) of all possible pairs of words in their names. The linguistic matching rule does not indicate the exact semantic relationship between the two concepts, for example, whether they have a class-subclass relationship, or are equivalent etc.

inference rules An inference engine in Datalog is capable of making logical derivations based on the inference rules available in the engine.

The ontologies that were used for the experiments were represented in RDF and contained 30 respective 50 nodes, which are very small ontologies. The authors demonstrate how the articulation rules are generated by the ONION system. The tool was evaluated by computing precision and recall measures for the corpus and thesaurus based word relators. Accuracy was measured by comparing the results of the automated articulation generator with those expected by the expert. If the expert deleted a match of the articulation generator then precision is lowered. In case the expert added a match that was not found by the articulation generator then recall is lowered.

The thesaurus-based method resulted in very poor results, thought the corpus-based method produced better results. However scalability was extremely low and the quality of the results were very dependent on the quality of the corpus available. When everything was pre-computed, the corpus-based method scaled very well.

8.19 Alimo (ITI-CERTH)

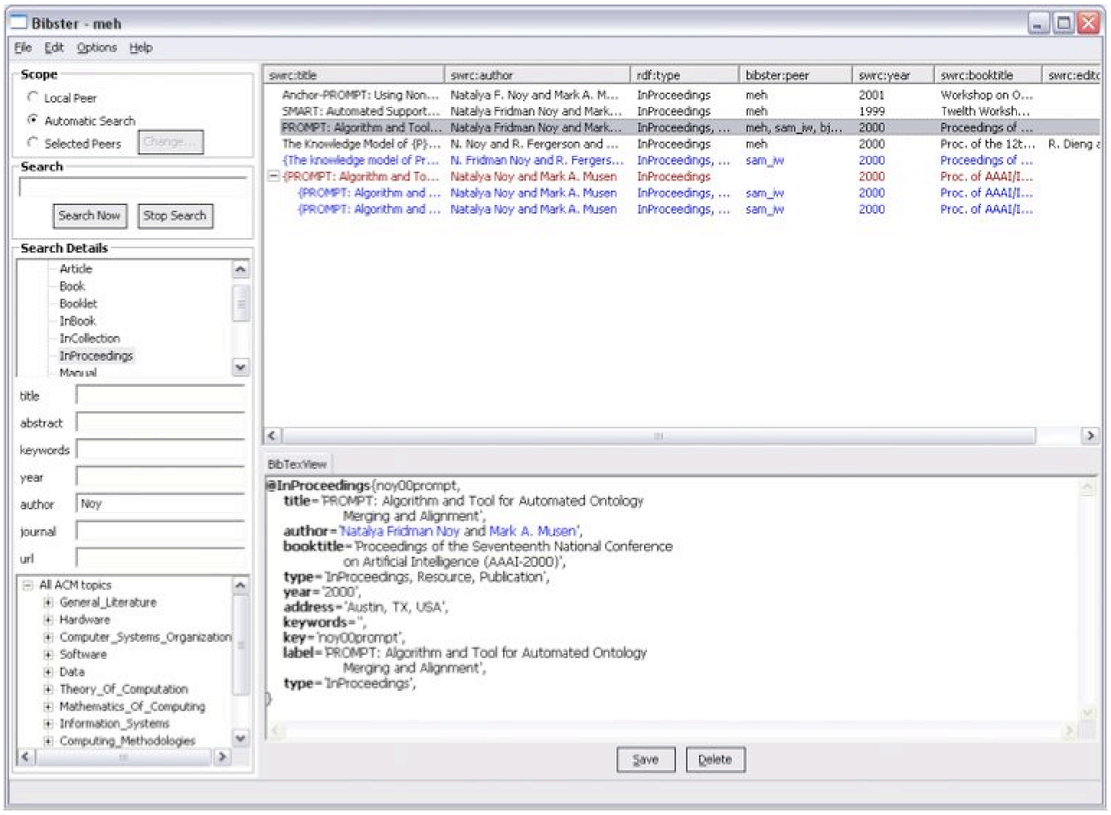
The development and maintenance of large multimedia databases has attracted much attention nowadays from companies and organizations that held multimedia content (archives, broadcast companies, radio and TV channels etc). The goal is to bypass the ineffective and time-consuming process of manual searching and retrieval of multimedia content and use computers to make the content easy to be found and accessible to other parties. Thus, two critical points are identified in making the above goal a reality; effective representation as well as effective retrieval and exploration of multimedia content. For accomplishing the above goal researchers have started to use ontologies in the field of multimedia in order to construct machine-understandable, descriptive versions of the multimedia content based on multimedia ontologies. Four different levels of information are represented in multimedia ontologies: signal information, featural information, symbolic information and semantic information.

With the aid of multimedia ontologies the vision of querying and retrieving multimedia content from distributed databases has started to become more feasible. But in order for someone to be able to use all the levels of information, from the semantic to the raw audiovisual one, a proper alignment framework should be provided. For this reason ITI-CERTH is constructing ALIMO (Alignment of Multimedia Ontologies), an ontology alignment system that pay special care to each one of the subparts of a multimedia ontology and the attributes with the special meaning and structure. Semantic descriptions will be aligned using methods hybrid alignment systems (terminological, structural etc). The signal description parts will be compared by using visual matching algorithms from the field of digital image and video processing. The feature description by examining the XML schema of the MPEG-7 visual part and at last the symbolic description by referring to the definitions of the concepts that those labels are instances of, and also by examining the datatypes of the attributes assigned to those instances.

8.20 Bibster (U. Karlruhe)

Bibster[3] [Broekstra *et al.*, 2004] addresses a typical problem in the daily life of a computer scientist, where one regularly has to search for publications or their correct bibliographic metadata. The scenario that we support here is that researchers in a community share bibliographic metadata in a Peer-to-Peer fashion.

Bibster is a Peer-to-Peer system based on the SWAP architecture[4], which allows to easily integrate, share and search bibliographic metadata using semantic technologies for the representation of the bibliographic instances and the peers' expertise to allow effectively route queries. Semantic similarity measures identifying duplicates allow to visualize and to integrate the heterogeneous search results from the peers. Bibliographic entries are extracted from BibTex into an ontology. The query results themselves represent small ontologies, containing duplicates.



author: Noy

swrc:title	swrc:author	rdf:type	bibster:peer	swrc:year	swrc:booktitle	swrc:editc
Anchor-PROMPT: Using Non...	Natalya F. Noy and Mark A. M...	InProceedings	meh	2001	Workshop on O...	
SMART: Automated Support...	Natalya Fridman Noy and Mark...	InProceedings	meh	1999	Twelth Worksh...	
PROMPT: Algorithm and Tool...	Natalya Fridman Noy and Mark...	InProceedings, ...	meh, sam_jw, bj...	2000	Proceedings of ...	
The Knowledge Model of {P}...	N. Noy and R. Fergerson and ...	InProceedings	meh	2000	Proc. of the 12t...	R. Dieng a
{The knowledge model of Pr...	N. Fridman Noy and R. Fergers...	InProceedings, ...	sam_jw	2000	Proceedings of ...	
{PROMPT: Algorithm and To...	Natalya Noy and Mark A. Musen	InProceedings		2000	Proc. of AAAI/I...	
{PROMPT: Algorithm and ...	Natalya Noy and Mark A. Musen	InProceedings, ...	sam_jw	2000	Proc. of AAAI/I...	
{PROMPT: Algorithm and ...	Natalya Noy and Mark A. Musen	InProceedings, ...	sam_jw	2000	Proc. of AAAI/I...	

```
@InProceedings{noy00prompt,
   title='PROMPT: Algorithm and Tool for Automated Ontology
            Merging and Alignment',
   author='Natalya Fridman Noy and Mark A. Musen',
   booktitle='Proceedings of the Seventeenth National Conference
            on Artificial Intelligence (AAAI-2000)',
   type='InProceedings, Resource, Publication',
   year='2000',
   address='Austin, TX, USA',
   keywords='',
   key='noy00prompt',
   label='PROMPT: Algorithm and Tool for Automated Ontology
            Merging and Alignment',
   type='InProceedings',
}
```

Figure 8.2: Screenshot of the Bibster Application

Finding duplicates is closely related to finding corresponding mappings. In both cases it is necessary to recognize identical objects despite their different identifiers.

In the given scenario duplicates are bibliographic entries which refer to the same publication or person in the real world, but are modelled as different resources. The similarity function is based on different features of the respective instances. For persons one can refer to the name. For pub-

[3]http://bibster.semanticweb.org
[4]http://swap.semanticweb.org

lications to title, authors, editors, journal, address, type of publication, etc. The function returns a value between 0 and 1 by applying specific heuristics to every feature: Strings are compared using the Levenshtein distance [Levenshtein, 1966], the authors of publications are compared by comparing the two sets. Some domain specific features require special heuristics: if the type of one publication is "Misc", this only means that no further information about the type was available. If another publication is e.g. type "Article" the similarity is set to 0.5 rather than 0. Besides individual functions our approach focuses on applying an aggregation function to achieve an overall similarity. Through transitive closure we receive a set of "identical" entities. Instead of presenting all instances of the query result, duplicates are visualized as one, merged, resource. These merged resources consist of a union of properties of the individuals identified as duplicates.

After several rounds of testing Bibster is now openly available [5], with the component based on alignment working in the background of the system.

8.21 QOM (U. Karlsruhe)

QOM considers both the quality of mapping results as well as the run-time complexity. The hypothesis is that mapping algorithms may be streamlined such that the loss of quality (compared to a standard baseline) is marginal, but the improvement of efficiency is so tremendous that it allows for the ad-hoc mapping of large-size, light-weight ontologies. To substantiate the hypothesis, a number of practical experiments were performed.

The outcome is QOM — Quick Ontology Mapping [Ehrig and Staab, 2004]. It is defined by the steps of a process model as shown in Figure 8.3. Mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O_1, we try to find a corresponding entity, which has the same intended meaning, in ontology O_2.

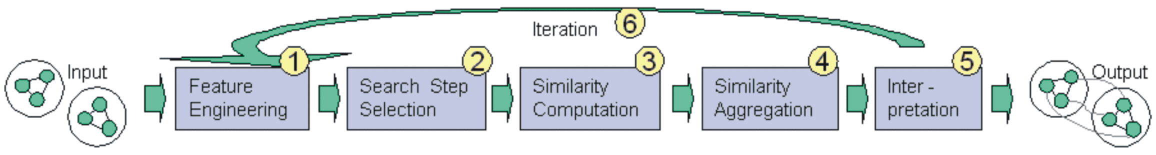


Figure 8.3: QOM Mapping Process

1. Firstly, QOM uses RDF triples as features.

2. Second, instead of comparing all entities of the first ontology with all entities of the second ontology, QOM uses heuristics to lower the number of candidate mappings, which is a major problem for run-time complexity. In this dynamic programming approach we only choose promising candidate mappings.

3. The actual similarity computation is done by using a wide range of similarity functions [Ehrig and Staab, 2004]. An entity is described by the kind of appearance that is found to hold for this entity for characteristics like: identifiers such as URIs, RDF/S primitives such as subclass and instance relations, or domain specific features e.g. a *hashcode-of-file*

[5]http://bibster.semanticweb.org

in a file sharing domain. These features of ontological entities are compared using *String Similarity* and *SimSet* for set comparisons. For efficiency reasons the similarity computation was disburdened by removing extremely costly feature-measure combinations such as the comparison of all subclasses of two concepts.

4. These individual measures are all input to the similarity aggregation. Instead of applying linear aggregation functions, QOM applies a sigmoid function, which emphasizes high individual similarities and de-emphasizes low individual similarities.

5. From the similarity values we derive the actual mappings. A threshold to discard spurious evidence of similarity is applied. Further mappings are assigned based on a greedy strategy that starts with the largest similarity values first.

6. Through several iteration rounds the quality of the results rises considerably. Eventually, the output returned is a mapping table representing the relation map_{O_1,O_2}.

The do2002a was very promising. Depending on the scenario QOM reaches high quality mapping levels very quickly. QOM is on a par with other good state-of-the-art algorithms concerning the quality of proposed mappings, while outperforming them with respect to efficiency — in terms of run-time complexity ($O(n)$ instead of $O(n^2)$) and in terms of the experiments we have performed (by a factor of 10 to 100).

8.22 KILT (INRIA Lorraine)

A short description of KILT, a maintenance tool for comparing knowledge base versions within the KASIMIR system (see [d'Aquin *et al.*, 2003]).

The KASIMIR system is a knowledge-based system aimed at helping the decision process when searching for an adequate treatment for patients ill with cancer. During an update (or a revision) of a KASIMIR knowledge base, the need for automatically comparing the old base KBold (before the update) and the new base KBnew (after the update) has appeared and is rather important for controlling the evolution of a knowledge base. A module comparing versions has to indicate what has been actually updated, and to check whether the modifications are in accordance with the intents of the knowledge engineer. This is the role of the module called KILT, that has been implemented and integrated into the PROTEGE knowledge editor. KILT enables to make a partitioning of the problems (i.e. a problem is described by a concept denoting a set of patients, and is possibly associated with a solution or a treatment), represented in KBold and/or KBnew in four parts:

1. The problems that appear in the two bases, with the same solutions;
2. The problems that appear in the two bases, with different solutions;
3. The obsolete problems, appearing in KBold but not in KBnew;
4. The new problems, appearing in KBnew but not in KBold.

The above partitioning is based on the use of the KASIMIR reasoner. For example, the new problems in category (4) can be found in the following way. Each problem PBnew of KBnew is classified in the hierarchy of KBold, which enables to check whether there is a problem PBold of KBold equivalent to PBnew, i.e. PBold subsumes and is subsumed by PBnew. If this is not the case, then PBnew is a new problem. The three other categories of problems (1), (2), and (3), can be detected and checked in a similar way. This shows that the implementation of KILT is rather simple, once the connection with the KASIMIR reasoner is done.

KILT is integrated in PROTEGE in the following way. During a session, KBold corresponds to the state of the knowledge base at the beginning of the session, and KBnew to its current state. Therefore, the KILT module enables to visualize the edition modifications, i.e. addition or removal of a problem, and association of another solution to an already known problem, at any time of the session. KILT makes comparisons at a semantic level: two concepts match when they have equivalent definitions, based on their attribute values and on the subsumption relation between classes. One main drawback is that it is assumed that the attributes –and their names– do not change from one knowledge base version to another.

Chapter 9

Conclusions

As as been demonstrated by the number of provided of use cases (§4) and developed systems (§8, alignments are useful in the semantic web. There have been a number of techniques on which they could be based, both for the local grouping of objects (§6) and the global computation (§7 of the alignment.

However, there is no common understanding of what to align, how to provide the results and what is important. But, it seems necessary in order to stimulate and integrate research to:

– be able to compare the approches;
– be able to combine them.

For that purpose, developing a common framework and benchmarking experimentation is a first answer to these two requirements. These are the topics of other deliverables (D2.2.1 and D2.2.2 respectively) on which the work is already ongoing.

Chapter 10

The Dublin Algorithm for Ontology Alignment

Most mapping algorithms adhere to a simple structure: an initial calculation of an intrinsic similarity measure is followed by an iterative calculation of an extrinsic (structural) measure, before finally the mappings are derived from the pairwise similarities. Our algorithm follows this common structure, too. However, there are two features which make it distinct from other algorithms that we are aware of. First, we compute the structural similarity by using a feature vector representation of each concept. Section 10.2 describes the details. Second, the way how the similarities are transformed into mappings differs from most current approaches. While Melnik et al. in [Melnik *et al.*, 2002] propose to compute either a stable marriage or the maximum weighted matching over a bipartite graph that represents the pairwise similarities of concepts, it seems that most newer ontology mapping algorithms do not do this (e.g. Ehrig and Staab use a simple greedy approach in [Ehrig and Sure, 2004]). In section 10.3.1 we describe how these two well-known graph algorithms can be used.

10.1 Computing Intrinsic Similarity

We use URIs, labels, comments and text from individuals and property values as text sources. In our implementation, we use distance metrics from the well-known SecondString library[1] as intrinsic similarity measures. We used a version of Levenshtein edit distance [Levenshtein, 1966] that is scaled to the range $[0, 1]$ for comparing labels and local names. We used a soft-token metric for comparing comments and instance data. To determine the overall intrinsic similarity between two concepts, we use the maximum of these metrics. To avoid overemphasizing small similarities, we disregard similarities that are smaller than a threshold of 0.4 and map similarities greater than 0.4 to the full range $[0, 1]$.

10.2 Computing Extrinsic Similarity

To compute the extrinsic similarity, we use a vector representation $\vec{\mathrm{de}}(v)$ for each entity and then compute the similarities between these vectors. To formally define the extrinsic feature vector, we

[1] `http://secondstring.sourceforge.net/`, see also [Cohen *et al.*, 2003]

first introduce a function that computes all entities that are connected to an entity v by a relation l.

Definition 36. *We define a function from the set of vertices and the set of labels L to the power set of vertices so that for a given vertex the function finds all vertices adjacent through an arc with a given label:*

$$\mathrm{rel} : V \times L \to 2^V$$

Let $G = (V, A)$ be a digraph with the set of vertices V and labelled arcs A as a set of ordered triples $(v, w, l) \in V \times W \times L$. Then we define:

$$\mathrm{rel}(v, l) = \{x | v, x \in V \wedge (v, x, l) \in A\}$$

The definition of $\mathrm{rel}' : V' \times L \to 2^{V'}$ *is analogous.*

Next, as an intermediate step to our extrinsic feature vector function, we define a *dynamic intrinsic* feature vector function as a vector representation of all similarities between an entity v and all entities $v' \in V'$. Dynamic intrinsic means that these features are inherent to an entity, but they are dynamic in the sense that their value can change as we get more information about that entity and can thus make a better prediction about the similarities between this and other entities. Note that the dynamic intrinsic features are what we want to compute. In particular, this means that the dynamic intrinsic features are initially unknown.

Definition 37. *We define a dynamic intrinsic feature vector function as:*

$$\vec{\mathrm{di}} : V \to \mathbb{R}^{|V'|}$$

Analogous to the matrix representation of a graph, we impose an arbitrary total order on V' and denote the first element of V' as v'_0 and the subsequent elements as v'_n for all $n < |V'|$. Then we define $\vec{\mathrm{di}}$ *as follows:*

$$\vec{\mathrm{di}}(v) = [\mathrm{sim}(v, v'_0), \mathrm{sim}(v, v'_1), \ldots, \mathrm{sim}(v, v'_{|V'|-1})]$$

Dynamic extrinsic features are dynamic intrinsic features of related entities:

Definition 38. *We define a dynamic extrinsic feature vector function as:*

$$\vec{\mathrm{de}} : V \to \mathbb{R}^{|V'|}$$

Assuming a commutative and associative operator $\oplus$ on $\mathbb{R}^d$ and a function rel *as per definition 36, we define* $\vec{\mathrm{de}}(v)$ *as some combination $\oplus$ of the dynamic intrinsic features* $\vec{\mathrm{di}}(x)$ *(see definition 37) of all related entities $x \in \mathrm{rel}(v)$.*

$$\vec{\mathrm{de}}(v) = \bigoplus_{x \in \mathrm{rel}(v)} \vec{\mathrm{di}}(x)$$

Note that the elements in $\vec{\mathrm{de}}(v)$ are based on the relations of $v \in V$, but correspond to vertices in V'. In order to compute an extrinsic similarity between v and some v', we have to define an extrinsic feature vector for v' that is based on the relations of $v' \in V'$.

Definition 39. *We define an extrinsic feature vector function as:*

$$\vec{\mathrm{de}'} : V' \to \mathbb{R}^{|V'|}$$

Based on the total order on V' from definition 37, we define that each element i in $\vec{\mathrm{de}'}$ *is* 1*, if $v'_i \in \mathrm{rel}(v')$ and* 0 *otherwise.*

Algorithm 1 Iterative Similarity Calculation

```
for $v \in V$ do
    $\vec{\mathrm{di}}_{\mathrm{int}}(v) \leftarrow [\mathrm{sim}_{\mathrm{int}}(v, v'_0), \mathrm{sim}_{\mathrm{int}}(v, v'_1), \ldots, \mathrm{sim}_{\mathrm{int}}(v, v'_{|V'|-1})]$
end for
$\vec{\mathrm{de}}(v) \leftarrow \bigoplus_{x \in \mathrm{rel}(v)} \vec{\mathrm{di}}_{\mathrm{int}}(x)$ {Initially, use intrinsic similarity only}
for a fixed number of iterations do
    for $v \in V$ do
        $\vec{\mathrm{di}}_{\mathrm{ext}}(v) \leftarrow [\mathrm{sim}_{\mathrm{ext}}(v, v'_0), \mathrm{sim}_{\mathrm{ext}}(v, v'_1), \ldots, \mathrm{sim}_{\mathrm{ext}}(v, v'_{|V'|-1})]$
        $\vec{\mathrm{di}}(v) \leftarrow \vec{\mathrm{di}}_{\mathrm{int}}(v) \otimes \vec{\mathrm{di}}_{\mathrm{ext}}(v)$ {Combine intrinsic and extrinsic similarity}
    end for
    $\vec{\mathrm{de}}(v) \leftarrow \bigoplus_{x \in \mathrm{rel}(v)} \vec{\mathrm{di}}(x)$
end forreturn $\forall v \in V : \vec{\mathrm{di}}(v)$
```

Given definitions 38 and 39 we can now easily define an extrinsic similarity function $\mathrm{sim}_{\mathrm{ext}}(v, v')$ based on the similarity between the vectors $\vec{\mathrm{de}}(v)$ and $\vec{\mathrm{de}}'(v')$. A common similarity measure for two vectors is the dot product, but it is usually better to normalize the similarity measure using the well-known cosine, Dice, Jaccard or overlap coefficients, which are widely used in information retrieval, e.g. [van Rijsbergen, 1975] or [Salton, 1989].

The similarities based on the extrinsic feature vectors are not symmetric. Since the feature vector is based on the best mapping for each concept, the fact that v maps to v' does not necessarily mean that the best mapping for v' is v, if the overall similarity $\mathrm{sim}(v, v')$ is greater than the similarity of v to all other $x' \in V'$ but less than the similarity $\mathrm{sim}(v', x)$ of v' to some $x \in V$.

10.3 Iterative Algorithm

Algorithm 1 formally specifies the iterative method of calculating the overall similarity. We are not restricted to computing $\mathrm{sim}(v, v')$, calculating $\mathrm{sim}(v', v)$ is analogous. Recall that because of the way the extrinsic similarity is defined they are not necessarily equal. The next section explains a way to exploit this asymmetry.

This algorithm is in fact very similar to the supervised learning algorithm that we presented in [Heß and Kushmerick, 2004] and could be seen as a generalization thereof. For that reason it is straightforward to incorporate background knowledge, e.g. a mapping to a third ontology that is known a priori, if we substitute a machine learning algorithm instead of a string distance metric. We will explore this possibility in future work.

10.3.1 Postprocessing Steps

Once we have computed the overall similarities, we have to compute the actual one-to-one mapping. This is the problem of finding a matching in a bipartite graph. A bipartite graph $B = (V + V', E)$ is a graph where the nodes can be split in two groups such that every edge connects two nodes from both partitions. Every similarity that has been calculated in the previous step corresponds to a weighted edge in such a bipartite graph. A matching M in a graph is a set of edges such that no node is incident to more than one edge. In our setting this corresponds to a one-to-one mapping: For every instance in one ontology we want to find one instance in the other ontology. M is called maximum-weighted, if there is no other matching where the sum of all edge weights in the matching is bigger. M is called a stable marriage, if there are no nodes $v \in V$ and $v' \in V'$ such that the edge between v and v' in B is not in M, but has a higher weight than the edges in M that are incident in v and v'. We used the Gale/Shapley algorithm [Gale and Shapley, 1962] to compute stable marriages and Munkres' algorithm [Munkres, 1957] (also referred to as the Hungarian algorithm) to compute maximum-weighted matchings.

The mappings submitted to the OAEI do2002a were computed with a fixed number of 5 iterations for the similarity calculation and using Munkres' algorithm to compute a maximum-weighted matching, which performed better than a setup with a stable marriage.

Chapter 11

oMAP: An Implemented Framework for Automatically Aligning OWL Ontologies

Ontologies are usually seen as a solution to data heterogeneity on the web [Euzenat and Valtchev, 2004]. An ontology is a way of describing the world: it allows to determine what kinds of things there are in the world, their characteristics, the relationships between them and more complex axioms. Since a lot of efforts are deployed to provide hands-on support for developers of Semantic Web applications, with the online publishing of "best practices", it is expected now that more and more ontologies covering partially the same subjects will be available on the web. Indeed, this is already true for numerous complex domains such that the medical or the multimedia domain. In such a case, some entities can be given different names or simply be defined in different ways or in different languages. The semantic interoperability has then to be grounded in ontology reconciliation. The underlying problem is often called the "ontology alignment" problem [Euzenat and Valtchev, 2004].

We focus here on ontologies described in the same knowledge representation language (OWL) and we propose a general framework named *oMAP* that aims to automatically align two OWL ontologies. *oMAP* [Straccia and Troncy, 2005b; Straccia and Troncy, 2005a] allows to find the best mappings (together with their weights) between the entities defined in the ontologies, using the prediction of several classifiers. These classifiers are terminological or machine learning-based, and we introduce a new one, that uses the semantics of the OWL axioms for establishing equivalence and subsumption relationships between the classes and the properties defined in the ontologies. *oMAP* can be downloaded for free [1].

Our approach is inspired by the data exchange problem [Fagin *et al.*, 2003] and borrows from others, like GLUE [Doan *et al.*, 2003b], the idea of using several specialized components for finding the best set of mappings. Theoretically, an ontology mapping in *oMAP* is a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ and $\mathbf{T}$ are respectively the source and target ontologies, and Σ is a finite set of *mapping constraints* of the form:

$$\alpha_{i,j}\, T_j \;\leftarrow\; S_i$$

[1] `http://homepages.cwi.nl/simtroncy/oMAP/`

where S_i and T_j are respectively the source and target entities. The intended meaning of this rule is that the entity S_i of the source ontology is mapped onto the entity T_j of the target ontology, and the confident measure associated with this mapping is $\alpha_{i,j}$. Note that a source entity may be mapped onto several target entities and conversely. But, we do not require that we have a mapping for every target entity.

Aligning two ontologies in *oMap* consists of three steps:

1. We form a possible Σ, and estimate its quality based on the quality measures for its mapping rules;
2. For each mapping rule $T_j \leftarrow S_i$, we estimate its quality $\alpha_{i,j}$, which also depends on the Σ it belongs to, i.e. $\alpha_{i,j} = w(S_i, T_j, \Sigma)$;
3. As we cannot compute all possible Σ (there are exponentially many) and then choose the best one, we rather build iteratively our final set of mappings Σ using heuristics.

Similar to GLUE [Doan *et al.*, 2003b], we estimate the weight $w(S_i, T_j, \Sigma)$ of a mapping $T_j \leftarrow S_i$ by using different classifiers $CL_1, \ldots, CL_n$. Each classifier CL_k computes a weight $w(S_i, T_j, CL_k)$, which is the classifier's approximation of the rule $T_j \leftarrow S_i$. For each target entity T_j, CL_k provides a rank of the plausible source entities S_{i_k}. Then we rely on a priority list on the classifiers, $CL_1 \prec CL_2 \prec \ldots \prec CL_n$ and proceed as follows: for a given target entity T_j, select the top-ranked mapping of CL_1 if the weight is non-zero. Otherwise, select the top-ranked mapping provided by CL_2 if non-zero, and so on.

In the next section, we briefly present the classifiers that are currently used in our framework. It is worth noting that some of them consider the terminological part of the ontologies only, while others are based on their instances (i.e. the values of the individuals). Finally, we end this section by introducing a new classifier that fully uses the structure and the semantics of ontology definitions and axioms.

11.1 Terminological, Machine Learning-based and Structural Classifiers

The terminological classifiers work on the name of the entities (class or property) defined in the ontologies. In OWL, each resource is identified by a URI, and can have some annotation properties attached. Among others, the `rdfs:label` property may be used to provide a human-readable version of a resource's name. Furthermore, multilingual labels are supported using the language tagging facility of RDF literals. In the following, we consider that the name of an entity is given by the value of the `rdfs:label` property or by the URI fragment if this property is not specified. The typical terminological classifiers we used in *oMAP* compare the name of the entities, their stem (using the Porter stemming algorithm [Porter, 1980]), compute some similarity measures between the entity names (once downcased) such that the Levenshtein distance[Levenshtein, 1966] (or edit distance), or compute a similarity measure between the entity names using the WordNet®[2] relational dictionary.

[2]`http://wordnet.princeton.edu/`

Additionally, an ontology often contains some individuals. It is then possible to use machine learning-based classifiers to predict the weight of a mapping between two entities. The instances of an OWL ontology can be gathered using the following rules: we consider (i) the label for the named individuals, (ii) the data value for the datatype properties and (iii) the type for the anonymous individuals and the range of the object properties. For example, using the abstract syntax of [Horrocks *et al.*, 2003], let us consider the following individuals :

```
Individual (x1 type (Workshop)
   value (label "Italian Semantic Web Workshop")
   value (location x2))
Individual (x2 type (Address)
   value (city "Trento") value (country "Italy"))
```

Then, the text gathered u_1 for the named individual x_1 will be (`"Italian Semantic Web Workshop", "Address"`) and u_2 for the anonymous individual x_2 (`"Address", "Trento", "Italy"`). Typical and well-known classifiers used in machine learning such as Naive Bayes and kNN [Sebastiani, 2002] have then been implemented in *oMAP* using these data.

Finally, we have drawn a new classifier which is able to use the semantics of the OWL definitions while being guided by their syntax. This *structural classifier* is fully described in [Straccia and Troncy, 2005b; Straccia and Troncy, 2005a]. It is used in the framework *a posteriori*. Indeed, we rely on the classifier preference relation $CL_{Name} \prec CL_{Stem} \prec CL_{EditDistance} \prec CL_{NaiveBayes}$. According to this preference relation, a set Σ' of mappings is determined. This set is given as input to the structural classifier. Then the structural classifier tries out all alternative ways to extend Σ' by adding some $T_j \leftarrow S_i$ if no mapping related to T_j is present in Σ'.

All the classifiers detailed previously have been implemented to be compatible with the alignment API [Euzenat, 2004], thus easing their chaining. Therefore, our *oMAP* framework benefits from all the do2002a facilities for comparing our approach with other methods. The problem of aligning ontologies has indeed already produced some interesting works. However, it is difficult to compare theoretically the various approaches proposed since they base on different techniques. Hence, it is necessary to compare them on common tests. This is the goal of the Ontology Alignment Evaluation Initiative (OAEI[3]) who set up do2002a campaign and benchmark tests for assessing the strengths and weakness of the available tools. We have evaluated *oMAP* with the data of the EON 2004 contest [Sure *et al.*, 2004] and we have participated actively to the 2005 campaign [Straccia and Troncy, 2005c].

11.2 Conclusion

As the number of Semantic Web applications is growing rapidly, many individual ontologies are created. The development of automated tools for ontology alignment will be of crucial importance. We have designed *oMAP*, a formal framework for ontology alignment, to cope this problem. *oMAP* uses different classifiers to estimate the quality of a mapping. Novel is the classifier which uses the structure of the OWL constructs and thus the semantics of the entities

[3]`http://oaei.inrialpes.fr`

defined in the ontologies. Furthermore, machine learning-based classifiers are employed. We have implemented the whole framework and evaluated it on independent benchmark tests provided by the Ontology Alignment Evaluation Initiative campaign.

As future work, we see some appealing points. Additional classifiers using more terminological resources can be included in the framework, and are currently under implementation while the effectiveness of the machine learning part could be improved using other measures like the kNN classifier or the KL-distance. While to fit new classifiers into our model is straightforward theoretically, practically finding out the most appropriate one or a combination of them is quite more difficult. In the future, more variants should be developed and evaluated to improve the overall quality of *oMAP*.

Chapter 12

Aligning Ontologies with Falcon

12.1 Overview

As an infrastructure for semantic web applications, Falcon[1] is a vision of our research group. It will provide enabling technologies for **F**inding, **A**ligning and **L**earning ontologies, and ultimately for **C**apturing knowledge by an **ON**tology-driven approach. It is still under development in our group. As a component of Falcon, Falcon-AO is an automatic tool for aligning ontologies. It is dedicated to aligning web ontologies expressed in OWL DL.

The overview of the system architecture of Falcon-AO is depicted in Fig.1. There are two matchers integrated in the current version (version 0.4): one is a matcher based on linguistic matching for ontologies, called LMO; and the other one is a matcher based on graph matching for ontologies, called GMO. The integration of the alignments generated by the two matchers is determined by the linguistic and structural comparability.

The main aligning process is outlined as follows:

1. Input two ontologies and parse them.
2. Observe the linguistic and structural comparability. In the case that both comparability are very low, the two ontologies are considered as totally different and Falcon-AO exits with no alignment.
3. Run LMO and obtain some alignments.
4. Set external entities of the ontologies according to the existing mapping pre-assigned by the system and the alignments generated by LMO.
5. Run GMO and obtain some additional alignments.
6. Integrate the alignments generated by LMO and GMO according to the linguistic and structural comparability.
7. Output the final alignments and exit.

[1]`http://xobjects.seu.edu.cn/project/falcon/falcon.htm`

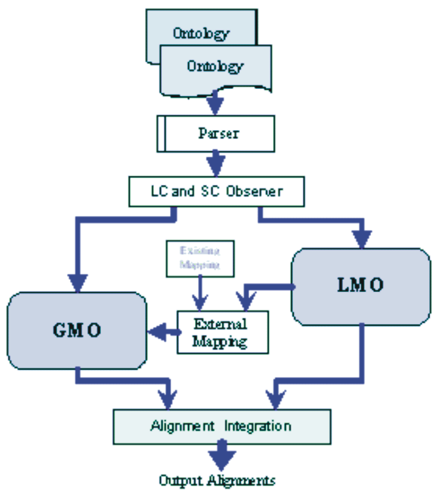


Figure 12.1: System Architecture

12.2 Specific Techniques

Three novel techniques are used in Falcon-AO. A brief introduction of these techniques are given in the following. More details can be found in [Hu *et al.*, 2005; Jian *et al.*, 2005; Qu *et al.*, 2006].

12.2.1 Linguistic Matching for Ontologies

LMO is based on an idea of virtual documents to pursue a cost-effective approach for linguistic matching. Basically, as a bag of weighted words, the virtual document of a URIref declared in an ontology contains not only the local descriptions but also the neighboring information to reflect the intended meaning of the URIref. Document similarity can be computed by traditional Vector Space techniques, and then be used in the similarity-based approaches to ontology matching.

12.2.2 Graph Matching for Ontologies

GMO uses bipartite graphs to represent ontologies, and measures the structural similarity between graphs. The idea of GMO is as follows: (a) similarity of two entities from two ontologies comes from the accumulation of similarities of involved statements (triples) taking the two entities as the same role (subject, predicate, object) in the triples; (b) the similarity of two statements comes from the accumulation of similarities of involved entities (including external entities) of the same role in the two statements being compared.

12.2.3 Linguistic vs. Structural Comparability

Falcon-AO integrates the matched entity pairs, which are generated by LMO and GMO, by observing the linguistic and structural comparability. The integration rules are described in brief as follows:

1. We take that linguistic comparability is somewhat more reliable than structural comparability, and that the alignments generated by LMO are always accepted by Falcon-AO.

2. When the linguistic comparability is high and the structural comparability is low, only alignments generated by GMO with high similarity are reliable and accepted by Falcon-AO.

3. If the linguistic comparability is low, all of the alignments generated by GMO are accepted by Falcon-AO. In this case, there is not enough information to measure these alignments and we can only assume that they are reliable.

12.3 Summary and Outlook

Falcon-AO is an automatic tool for aligning ontologies. Now, it integrates two matchers: LMO (A Linguistic Matching for Ontologies) and GMO (A Graph Matching for Ontologies). The experimental results on OAEI 2005 campaign demonstrate that Falcon-AO (version 0.3) performs very well on both Benchmark Test and Directory Test.

Some improvements will be considered in the future work: (a) the measurements of the linguistic and structural comparability of ontologies are still simple and an improvement will be needed, (b) the incorporation of corpus-based distributional similarity among words will be considered; and (c) some machine learning techniques will be integrated to realize a more powerful ontology matching tool.

Chapter 13

Ontology and Multimedia Ontology Alignment with ALIMO

In the effort to add multimedia documents in the Semantic Web multimedia ontologies will play an important role. In contrast to the usual ontologies, multimedia ontologies are formed by three different parts. The first part is the usual ontological part found in all web ontologies, which includes class, property and restriction definitions. The second part is the visual description part, where multimedia documents are given a visual description based on an MPEG-7 visual ontology. At last the third part is the actual raw data of the multimedia document. As it is obvious multimedia ontologies introduce new issues in task of (multimedia) ontology alignment that need to be tackled. For that purpose we are developing the platform ALIMO (Alignment of Multimedia Ontologies) which deals with all the features of multimedia ontologies.

The ALIMO platform consists of two matching modules. The first module is an ontology alignment method, which uses classical techniques for ontology alignment as the ones described in [Euzenat *et al.*, 2004a]. The second module consists of a visual matching algorithm.

13.1 Ontology Alignment Module

The ALIMO platform uses three types of matching methods. These are the following:

- **Terminological Matching:** This method computes the similarities based on the strings of class and property names.
- **Structural Internal Matching:** In this method we refine the similarity computed by terminological matching, for two classes, by a portion of the similarities between the names of their properties.
- **Structural External Matching:** In this method we refine the similarity between two classes by a portion of the similarity computed for the super-classes of two classes.

For the assessment of the similarity between two class or property names ALIMO uses a novel string matching algorithm, called I-Sub Matching. This algorithm [Stoilos *et al.*, 2005], is an extension of the well known Sub-String matching method towards several directions. First of all we believe that the similarity between two entities should be a function of both their commonalities as well as their differences. From that observation we have the following equation:

$$Sim(s_1, s_2) = Comm(s_1, s_2) - Diff(s_1, s_2) + winkler(s_1, s_2) \tag{13.1}$$

where $Comm(s_1, s_2)$ stands for the commonality between s_1 and s_2, $Diff(s_1, s_2)$ for the difference and $winkler(s_1, s_2)$ for the improvement of the result using the method introduced by Winkler in [Winkler, 1999]. Now, as a function of commonality we have used and extended the Substring distance metric. In contrast to the usual implementation, which searches only for the biggest common substring between two strings, we continue to find further common substrings until we have identified them all. Then we scale the length of the common substrings according to the following formula:

$$Comm(s_1, s_2) = \frac{2 * \sum_i length(maxComSubString_i)}{length(s_1) + length(s_2)} \tag{13.2}$$

As for the difference function, this is based on the length of the unmatched strings that have resulted from the initial matching step. Moreover, we believe that difference should play a less important role on the computation of the overall similarity. Our choice was the Hamacher product [Hamacher *et al.*, 1978], which is a parametric triangular norm. This leads us to the following equation:

$$Diff(s_1, s_2) = \frac{uLen_{s_1} * uLen_{s_2}}{p + (1 - p) * (uLen_{s_1} + uLen_{s_2} - uLen_{s_1} * uLen_{s_2})} \tag{13.3}$$

where $p \in [0, \infty)$, and $uLen_{s_1}$, $uLen_{s_2}$ represent the length of the unmatched substring from the initial strings s_1 and s_2 scaled with the string length, respectively.

Many ontology alignment algorithms use threshold values by which they determine which pairs of entities are to be considered similar and which not after a run of the algorithm. Obviously, the choice of the threshold is very crucial since a bad selection could remove many correct pairs or identify dissimilar ones as semantically equivalent. As pointed in [Stoilos *et al.*, 2005], one of the important features of the I-Sub method is that it improves the *stability* (threshold tolerance) of ontology alignment methods, compared to other string matching methods that exist in the literature. In other words, variations of the threshold of a platform from the optimal value will not affect the performance of the alignment platform, as is the case with most of the string matching methods.

In Figure 13.1 we can see our experimentation with ontology alignment using several popular string matching methods found in literature. The figure shows an average Recall versus average Precision chart relative to nine different threshold values used, ranging from 0.1 to 0.9. As we can see, all string matching methods achieve the best combination of precision and recall after the third/fourth threshold value (0.3/0.4). In terms of recall this can be interpreted to the interval from 0.8 to 0.83. From that point we can observe that if we increase (decrease) the threshold by one or two units we face a high degradation of the recall (precession), gaining in precision (recall). On

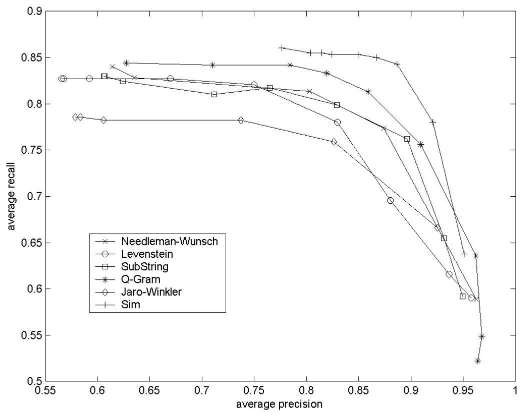


Figure 13.1: Average Precision vs. Average Recall values

the other hand the I-Sub method enjoys an area of 7 different threshold values, from 0.1 to 0.7, where precision can be increased, by increasing the threshold, while no or minor decrease in recall is encountered.

Chapter 14

FOAM – Framework for Ontology Alignment and Mapping

In recent years we have seen a range of research work on methods proposing alignments [Doan *et al.*, 2003a; Noy and Musen, 2003]. When one tries to apply these methods to some of the real-world scenarios of other research contributions [Ehrig *et al.*, 2003], one finds that existing alignment methods do not suit the given requirements: high quality results, efficiency, optional user-interaction, flexibility with respect to use cases, and easy adjustment and parametrization. The goal is to provide the end-user with a tool taking ontologies and returning alignments meeting these requirements. The Framework for Ontology Alignment and Mapping (FOAM[1]) itself consists of the general alignment process, specific extensions beyond its predecessor QOM, as presented in a previous deliverable, and pointers to the tool itself.

14.1 Alignment Process

One can observe that alignment methods like QOM [Ehrig and Sure, 2004] or PROMPT [Noy and Musen, 2003] may be mapped onto a generic alignment process (Figure 14.1). We refer to [Ehrig and Sure, 2004] for a detailed description. Here we will only mention the six major steps to clarify the underlying approach for the FOAM tool.

1. Feature Engineering, i.e. select excerpts of the overall ontology definition to describe a specific entity (e.g. label of an instance). FOAM makes use of all the features of OWL, including cardinality restrictions or enumeration definitions. Further domain-specific features may also be added.

2. Search Step Selection, i.e. choose two entities from the two ontologies to compare (e_1,e_2). Most approaches compare every entity of one ontology with every entity of the other ontology, but more efficient implementations are possible.

3. Similarity Assessment, i.e. indicate a similarity for a given description (feature) of two entities (e.g., $\text{sim}_{\text{superConcept}}(e_1,e_2)$=1.0).

[1]`http://www.aifb.uni-karlsruhe.de/WBS/meh/foam`

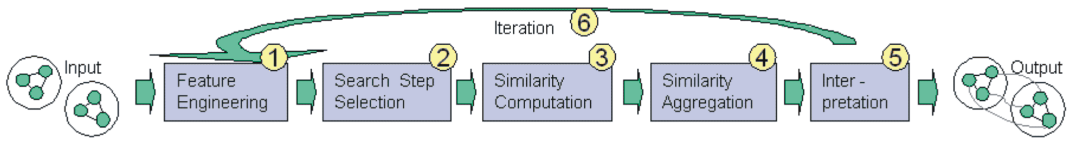


Figure 14.1: General Alignment Process

4. Similarity Aggregation, i.e. aggregate multiple similarity assessment for one pair of entities into a single measure.
5. Interpretation, i.e. use all aggregated numbers, a threshold and interpretation strategy to propose the alignment (align(e_1)='e_2').
6. Iteration, i.e. as the similarity of one entity pair influences the similarity of neighboring entity pairs, the equality is propagated through the ontologies.

Finally, we receive the alignments linking the two ontologies.

14.2 Extensions

Within the last year numerous additional methods extend the standard alignment process.

QOM – Quick Ontology Mapping: The QOM method [Ehrig and Sure, 2004] tackles the efficiency problem, which occurs when aligning larger ontologies. For this it makes use of the ontology structure. The number of candidate alignments to compare is considerably lowered by only allowing those which have very similar identifiers (or labels) or being a close neighbor of other existing alignments. Further, only those features are called on which do not require a complete traversing of the ontology, e.g., only direct instances of one concept are compared instead of all instances of all subconcepts. Both on theoretical and practical level the process is considerably sped up.

APFEL – Alignment Process Feature Estimation and Learning: Already the selection of which features to compare and which similarity measure to apply is very difficult. Setting aggregation weights for each feature is almost impossible, even for ontology experts. APFEL [Ehrig *et al.*, 2005] therefore is a method which solves these problems by using machine learning techniques. The user only has to provide some ontologies with known correct alignments. The learned decision tree is then used for aggregation and interpretation of the similarities.

Interactive Integration: So far the approaches focused on full-automation. However, it does make sense to include the user in the loop for some applications. By posing clever questions to the user he should be least bothered and at the same time receive best results. This is achieved by only presenting those candidate alignments to the user which are close to the threshold, i.e., for the system it is most uncertain whether they are true or false alignments. By manually tagging these accordingly quality of the results again increases considerably.

Adaptive Integration: The examination of several application scenarios [de Bruijn and Feier., 2005] has shown that the requirements for an alignment approach differ considerably, e.g., high efficiency versus high quality. FOAM has therefore been adapted once more. After the user has entered the scenario (alignment discovery, integration, merging, evolution, etc.) the parameters for the alignment process are chosen automatically [Ehrig and Sure, 2005]. This leads to better results, not in general, but for the specific scenario the alignments are required for. Thus, one implementation can be easily applied to several scenarios.

14.3 Implementation

The Framework for Ontology Alignment and Mapping (FOAM) has been implemented in Java. Further, it relies on the KAON2-environment[2] for processing ontologies (in specific ontologies represented in OWL-DL). This direct procedural approach can be very focused on specific problems arising for the alignment process, e.g., efficiency.

FOAM and its predecessors have been successfully applied in different applications. Within the SWAP-project,[3] FOAM was used to align and merge identical entities which were returned in the Bibster application or propose new aligned entities to the design board as needed in Xarop. Further, FOAM is a substantial part of the mediation component in the SEKT project.[4] Finally, the methods implemented in FOAM have been tested in three ontology alignment campaigns: I3CON, EON-OAC, and OAEI. FOAM behaved very favorable with results in the upper third of all systems, despite using only the standard full-automatic methods. Concrete results can be found in Part II.

FOAM is also an example of successful transition from research to industry. It has been integrated into the OntoMap tool, a graphical ontology mapping tool within the commercially sold OntoStudio framework of Ontoprise.[5]

The Framework for Ontology Alignment and Mapping is available through its webpage [6]. On the page one can find links to relevant publications, a download section of binaries and source code, installation guidelines and the documentation of FOAM, and some ontologies to test the tool. Further, there is a web-interface for internet users interested in very shallow testing. For real use is is recommend to download it.

[2]http://kaon2.semanticweb.org/
[3]http://swap.semanticweb.org/
[4]http://www.sekt-project.org/
[5]http://www.ontoprise.de/content/e3/e43/index_eng.html
[6]http://www.aifb.uni-karlsruhe.de/WBS/meh/foam

Chapter 15

OLA: OWL-Lite Alignment

OLA (for *O*WL-*L*ite *A*lignment) is an open-source tool jointly developed by teams at University of Montréal and INRIA Rhône Alpes. It features similarity-based alignment and a set of auxiliary services supporting the manipulation of alignment results [Euzenat and Valtchev, 2003; Euzenat and Valtchev, 2004].

Among the variety of alignment approaches (e.g., using machine learning, subsumption computation, formal concept analysis, etc.) similarity-based ones rely on a quantitative assessment of pair-wise likeness between entities. OLA, features a similarity model rooted in principles such as: completeness on the ontology language features, weighting of different feature contributions and mutual influence between related ontology entities. The resulting similarities are recursively defined hence their values are calculated by a step-wise, fixed-point-bound approximation process.

For the OAEI 2005 campaign, OLA was provided with an additional mechanism for weight determination that increased the autonomy of the system.

15.1 Overview

The primary goal behind the OLA tool design is to perform alignment of ontologies expressed in OWL, with a short-term emphasis on OWL-Lite and long-term one on OWL-DL. However, its GUI component, VISON[1] allows for many other services involving alignments (in the sense of [Euzenat, 2004]) to be accessed.

15.1.1 Functional specifications

From a mere algorithm for automated alignment construction, OLA has grown for the last year to an environment for alignment manipulation. Indeed, in its current version, the system offers, via its GUI component VISON, the following services:

- parsing and visualization of OWL-Lite and OWL-DL ontologies,
- computation of similarities between entities from two ontologies,
- extraction of alignments from a pair of ontologies, provided with a set of similarity matrices, one per category of ontology entities (see below),
- manual construction of alignments by composing entity pairs from two ontologies,

[1] `http://www.iro.umontreal.ca/`simowlola/

- use of an existing (partial) alignment as a seed for automated alignment construction (alignment completion),
- alignment visualization,
- comparison of two alignments.

In the remainder, the focus will be limited to the automated alignment construction with OLA.

15.1.2 Principles of matching in OLA

The following fundamental principles underly the design of the three key mechanisms in OLA – internal representation of the ontology, similarity computation and alignment extraction – that are involved in the global ontology alignment process:

All-encompassing comparison : We tend to believe that all the available knowledge about a pair of ontology entities should be taken into account when aligning. This does not exclude the possibility of ignoring particular aspects, i.g., OWL instances in case of OWL class comparison. However such a choice should be deliberately made by the tool user, here through appropriate weight assignment, or, if performed by an automated mechanisms, should reflect some particularity, either of the entire ontology (e.g., global absence of instances in both ontologies) or of the pair of entities at hand (e.g., local absence of instances in the pair of classes to be compared).

Highest automation level : Although we recognize that the entire alignment process often needs to be set on a semi-automated basis, we nevertheless argue in favor of a completely automated process for ”draft” alignment generation. Thus, we see the OLA user providing a minimal set of parameters at the initial steps of the process whereas the tool will suggest one or more candidate alignments at the end, without any other human intervention.

Category-dependent comparison : Following the syntactic structure of the OWL language, entities are divided into categories, e.g., *classes*, *objects*, *properties*, *relations*, and only entities of the same category are compared. Moreover, the entities of a category are compared using similarity functions of the same basic shape. The respective category functions comprise the same factors and the same weights. They are further customized for each pair of category entities by projecting them over the actual feature space of the entities (which may be far smaller than the complete space of the category).

Comparability of similarity results : To enable comparison of similarity scores between different alignment tasks but also for some computational reasons, a set of useful properties is insured for the similarity functions: *normalization*, *positiveness*, *maximalness*[2], and *symmetry*[3].

15.1.3 Current limitations

- Although it would be valuable for alignment, OLA currently offers no inference mechanisms that could help complete the entity descriptions. In particular, inheritance is not used to expand entities, mostly out of efficiency considerations.

- Although neighborhoods play crucial role in the similarity definition, two neighbor entities are not necessarily affecting each other’s respective similarities to a pair of other entities.

[2]With normalization, this amounts to forcing scores of 1 for identical entities within identical ontologies

[3]The price to pay for symmetry is the impossibility of detecting subsumption by this purely numerical procedure.

As only descriptive knowledge is taken into account, given two such entities, say e_1 and e_2, for e_2 to appear in a similarity expression for e_1, it should be considered as part of the description of the latter. For instance, a data type is not seen as being described by a property whose range the datatype represents. Consequently, datatypes are compared in an ontology-independent manner.

– Category borders are not similarity-permeable: Only entities from the same category are compared for similarity and hence for alignment.

15.1.4 Specific techniques used

OLA features an alignment process that splits into three basic steps: constructing the intermediate representation of the compared ontologies as labeled graphs, computing the similarity of each pair of same-category entities from the respective ontology graphs, extracting an alignment from the similarity matrices for each category.

15.1.5 OL-Graph construction

OL-Graphs are graph structures that provide an easy-to-process inner representation of OWL ontologies. An OL-Graph is a labeled graph where vertices correspond to OWL entities and edges to inter-entity relationships. As described in [Euzenat and Valtchev, 2004], the set of different vertex categories is: class (C), object (O), relation (R), property (P), property instance (A), datatype (D), datavalue (V), property restriction labels (L). Furthermore, we distinguish between datatype relations (R_{dt}) and object relations (R_o), and between datatype properties (P_{dt}) and object ones (P_o).

The OL-Graph model allows the following relationships among entities to be expressed:

– *specialization* between classes or relations (denoted $\mathcal{S}$),
– *instanciation* (denoted $\mathcal{I}$) between objects and classes, property instances and properties, values and datatypes,
– *attribution* (denoted $\mathcal{A}$) between classes and properties, objects and property instances;
– *restriction* (denoted $\mathcal{R}$) expressing the restriction on a property in a class,
– *valuation* (denoted $\mathcal{U}$) of a property in an object.

The OL-Graph of an ontology is built after the ontology is parsed[4]. The process of OL-Graph construction is described in [Tounazi, 2004].

15.1.6 Similarity model

The similarity functions used in OLA are designed in a category-specific manner and cover all the available descriptive knowledge about an entity pair. Thus, given a category X of OL-Graph nodes, the similarity of two nodes from X depends on:

– the similarities of the terms used to designate them, i.e., URIs, labels, names, etc.,
– the similarity of the pairs of neighbor nodes in the respective OL-Graphs that are linked by edges expressing the same relationships (e.g., class node similarity depends on similarity of superclasses, of property restrictions and of member objects),

[4]So far, we use the OWL API [Bechhofer *et al.*, 2003].

Funct.	**Node**	**Factor**	**Measure**
Sim_O	$o \in O$	$\lambda(o)$	sim_L
		$a \in A$, $(o, a) \in \mathcal{A}$	$MSim_A$
Sim_A	$a \in A$	$r \in R$, $(a, r) \in \mathcal{R}$	Sim_R
		$o \in O$, $(a, o) \in \mathcal{U}$	$MSim_O$
		$v \in V$, $(a, v) \in \mathcal{U}$	$MSim_V$
Sim_V	$v \in V$	value literal	type dependent
Sim_C	$c \in C$	$\lambda(c)$	sim_L
		$p \in P$, $(c, p) \in \mathcal{A}$	$MSim_P$
		$c' \in C$, $(c, c') \in \mathcal{S}$	$MSim_C$
sim_D	$d \in D$	$\lambda(r)$	XML-Schema
Sim_R	$r \in R$	$\lambda(r)$	sim_L
		$c \in C$, $(r, \texttt{domain}, c) \in \mathcal{R}$	$MSim_C$
		$c \in C$, $(r, \texttt{range}, c) \in \mathcal{R}$	$MSim_C$
		$d \in D$, $(r, \texttt{range}, d) \in \mathcal{R}$	Sim_D
		$r' \in R$, $(r, r') \in \mathcal{S}$	$MSim_R$
Sim_P	$p \in P$	$r \in R$, $(p, r') \in \mathcal{S}$	Sim_R
		$c \in C$, $(p, \texttt{all}, c) \in \mathcal{R}$	$MSim_C$
		$n \in \{0, 1, \infty\}$, $(p, \texttt{card}, n) \in \mathcal{R}$	equality

Table 15.1: Similarity function decomposition (`card` = `cardinality` and `all` = `allValuesFrom`).

– the similarity of other local descriptive features depending on the specific category (e.g., cardinality intervals, property types)

Datatype and datavalue similarities are external to our model and therefore they are either user-provided or measured by a standard function (e.g., string identity of values and datatype names/URIs).

Formally, given a category X together with the set of relationships it is involved in, $\mathcal{N}(X)$, the similarity measure $Sim_X : X^2 \to [0, 1]$ is defined as follows:

$$Sim_X(x, x') = \sum_{\mathcal{F} \in \mathcal{N}(X)} \pi^X_{\mathcal{F}} MSim_Y(\mathcal{F}(x), \mathcal{F}(x')).$$

The function is normalized, i.e., the weights $\pi^X_{\mathcal{F}}$ sum to a unit, $\sum_{\mathcal{F} \in \mathcal{N}(X)} \pi^X_{\mathcal{F}} = 1.$ for the computability The set functions $MSim_Y$ compare two sets of nodes of the same category (see [Euzenat and Valtchev, 2004] for details). Table 15.1 illustrates the set of similarities in our model.

OLA relies on various functions for identifiers comparison. Both string distances and lexical distances are used. Lexical distances rely on an exploration of WordNet 2.0 [Miller, 1995] with a quantitative assessment of the "relatedness" between two, possibly multi-word, terms. More specifically, the degree of relatedness between two WordNet entries is computed as the ratio between the depth, in graph-theoretic sense, of the most specific common hypernym and the average of both term depths. The computation of multi-word term similarity consists in first splitting the terms into a set of tokens each and then comparing all possible pairs of tokens from opposite sets using the above depth-based principle. The global term similarity is then computed as a similarity-based matching between both sets (see above).

As circular dependencies are impossible to avoid with the above definitions, the computation of the similarity values requires non-standard mechanisms. Following [Bisson, 1992; Valtchev, 1999], an equation system is composed out of the similarity definitions where variables correspond to similarities of node pairs while coefficients come from weights. The process of iterative, fixed-point-bound resolution of that system, as well as the related convergence and determinism issues are described in [Euzenat and Valtchev, 2004].

15.1.7 Implementation

OLA is implemented in JAVA. It is an implementation of the Alignment API [Euzenat, 2004] extending the standard implementation. OLA relies on the OWL API [Bechhofer *et al.*, 2003] for parsing OWL files. An entire subsystem is dedicated to the onstruction of OL-Graphs on top of the parsed ontologies. A set of further components that offer similarity computation services: substring distances, edit distances, Hamming distance, WordNet interface (via the JWNL library [Didion, 2004]), etc., that were originally designed for OLA are now part of the Alignment API. The VISON GUI component offers a uniform interface to all services provided by Alignment API and OLA. In particular, it visualizes both the input data, i.e., the OL-Graphs, and the final result, i.e., the alignment file, of the global process.

15.2 Improvements made for the 2004 do2002a

Several changes have been made to fit the complexity of the comparison. The most noteworthy one is the abandon of the requirement that all entities of the same category are compared along the same feature space.

15.2.1 Adaptive description space

Following the lessons learned with our participation in the EON 2004 alignment contest [Euzenat and Valtchev, 2004], we found that the "uniform factor weights" condition tends to favor pairs of entities that have complete descriptions, i.e., pairs where both the members are connected to at least one descriptive entity for each of the similarity factors in the respective formula. Conversely, pairs where a particular factor is void tend to score to lesser similarity values. The extreme case is the pair of `Thing` classes which, if present, usually have almost no description. With fixed weights for similarity factors, and hence universal feature space for comparison, the `Thing` class pair will be evaluated to a relatively weak similarity value and the chances are high for it to be skipped from the alignment.

Consequently, we have adapted the above measure to fit cases where particular pair of entities is described only by a small subset of the entire set of category descriptors. Thus, a descriptive factor is ignored for similarity computation whenever neither of the compared entities possesses a neighbor with the underlying link label (e.g., no instances for a pair of compared classes). In this case, not only its weight is set to 0, but also the weights of the remaining ”active” factors are increased correspondingly. To scale that principle up to the entire set of descriptive factors, the following simple mechanism has been realized in OLA: In order to keep both normalization and equity in similarity values, the weights of all non-null factors for a given entity pair are divided through their sum.

Thus, for a category X, the similarity measure $Sim_X^+ : X^2 \rightarrow [0, 1]$ becomes:

$$Sim_X^+(x, x') = \frac{Sim_X(x, x')}{\sum_{\mathcal{F} \in \mathcal{N}^+(x,x')} \pi_\mathcal{F}}$$

where $\mathcal{N}^+(x, x')$ is the set of all relationships $\mathcal{F}$ for which $\mathcal{F}(x) \cup \mathcal{F}(x') \neq \emptyset$ [5].

15.2.2 Lexical similarity measure

The initial straightforward similarity measure has been replaced by a more sophisticated one that better accounts for semantic proximity between compound identifiers. Thus, given a pair of identifiers, they are first “tokenized”, i.e., split into a set of atomic terms. Then, the respective pairs of terms are compared using WordNet. In fact, their degree of relatedness is computed as the ratio between the depth of the most specific common hypernym and the sum of both term depths. Finally, a similarity-based match is performed to establish a degree of proximity between the sets of terms.

15.2.3 Weight finding mechanism

To increase the level of automation in OLA, a weight-search mechanism was added to the initial architecture. Indeed, it is far from obvious for a novice user how to weight the different similarity factors. The underlying module performs several runs of the alignment producing subsystem with various weight combinations. It keeps only the combination that has resulted in the best alignment, i.e., the one of the highest total similarity between aligned entities. On the one hand, this procedure is not realistic in a setting where reference alignments are not given. On the other hand, if the tests a realistic, then what is learned is the best behaviour of the system in general.

15.3 Improvements made for the 2005 do2002a

Along the preparation of the OAEI 2005 campaign, a row of changes have been made to the system in order to make it fit the complexity of the alignment discovery task. The most striking one is the introduction of a weight-computing mechanism that eliminates the necessity for the tool user to provide initial weights and hence makes a significant step towards full automation of the alignment process.

[5] That is, there exists at least one y such that $(x, y) \in \mathcal{F}$ or at least one y' such that $(x', y') \in \mathcal{F}$.

15.3.1 Weight computing mechanism

As it is far from obvious for novice users how to weigh the different similarity factors, we initiated work on incorporating a weight computing mechanism within the system. The intended mechanism is both intuitive and effective so that alignment practitioners with various skill levels could find a match for their knowledge and experience. So far, we used a simple heuristic method that, according to the obtained results, performs reasonably well. The basic idea of the method consists in distributing the weights among similarity factors in the generic similarity function of a node category according to the relative importance of the corresponding category in the entire ontology. That is to say we use the average number of links of the corresponding type per entity of the category at hand. For instance, the greater the number of super-class links in the ontology, the higher the weight of the super-class factor in the class similarity formula.

15.3.2 Similarity measure for entity names

OLA uses two alternative modes of comparison for entity names (URIs, labels, etc.): a string measure[6] (a default) and a lexical similarity measure that relies on WordNet 2.0 (see above).

The highly sophisticated lexical similarity measure that was used in OLA for the EON competition has been replaced by a simpler but more purposeful one. Indeed, the initial function compared multi-word terms on three separate axes: nouns, verbs and adjectives, as provided by WordNet 2,0. Such comparison seemed appropriate for cases where the meanings of a word fall in more than one part-of-speech category. The inter-word similarities on each axis were aggregated by an independent best-match computations while the three resulting values were further combined to a single one via a weighted sum.

The new measure trades separate matchings on speech-part-wise basis to a single global matching along entry similarities that aggregate all three possible aspects of a word. Thus, the words are compared to each other with all possible meanings and the highest similarity over a single pair of meanings is taken for the words.

For the OAEI competition, as we had to rely on a fixed parameter set for the entire collection of tests, we have chosen to force the use of the string distance. Indeed, it showed better performances while being much more efficient than the WordNet-based computation.

Nevertheless, the improved lexical similarity was not completely discarded: it is currently used as a pre-processing tool that helps decide automatically the distribution of weights among similarity factors.

15.3.3 Minor adaptations

Following experiences from EON 2004, a set of simple but decisive modifications have been applied in order to prevent the precision leak in the tests. First, the instances have been excluded from the alignments by default, although the possibility is given to the user to reverse this choice. Then, entities external to the ontologies at hand have also been excluded from the alignment (but not from the similarity computation). Finally, one-to-one alignment production has been enforced in OLA to increase the potential recall of the resulting alignment.

[6]subString distance provided by the Alignment API

15.4 Results

The results obtained in the OAEI-2005 do2002a are grouped by test categories.

15.4.1 Tests 1XX

OLA performed very well on the tests of this group. This seems to be due to the fact that while the language varies along the individual tests of the group, the basic ontology entities involved in the similarity computation remain unchanged with respect to the reference ontology.

15.4.2 Tests 2XX

The performances of the algorithm seem to suggest that three sub-groups of tests can be distinguished. The first one comprises the tests 21X, 22X, 23X and 24X, with a small number of exceptions where the performance have been:

– Quite good: This is the case of tests 201, 202, with random class names. The random names were putting a strain on the ability of the algorithm to propagate similarity along the network of node pairs. Obviously, our technique needs some improvements on that point.

– Satisfactory: In the case of tests 248, 249, there is a combination of missing (or random) names with one other missing factor. For tests 248, 249, the missing factors are hierarchy (sub-class links) and instances, respectively. Both play important role in similarity computation of classes, whenever these are stripped of their names as is the case with these two ontologies. Hence the sharp drop in precision and recall with respect to the preceding tests.

– Weak: The notorious failure here have been the tests 205, 209, which are the only ones to use of synonymous names in the ontology entities (with respect to the intial ontology). As WordNet has been plugged-out of the similarity computation, these results are not surprising.

The second groups is made of the tests 25X. Here OLA performances varied substantially: from extremely poor (254) to satisfactory (252, 259).

The last five ontologies of the group, the 26X ones, have proven to represent a serious obstacle for OLA. The performances of the system here were poor to very poor.

15.4.3 Tests 3XX

The real-world ontologies of the group 30X made OLA perform in an unimpressive way. We believe that this is due to the fact that string similarity was systematically used as identifier comparison means. Indeed, tentative runs with WordNet as basis for name similarity yielded way more precise alignments on that group. Unfortunately, they also brought down the overall statistics from the entire test set such as mean precision and mean recall. Hence the choice of the WordNet-based lexical similarity for a default name comparison means has been temporarily dropped.

15.4.4 Directory tests

We are glad to won this test especially since it was blind. However, the low level of recall shows that there is room for improvement (note that OLA is rather targeting ontologies in expressive languages so this kind of tests is not its primary target). We did not analyse the causes of failure so far.

15.4.5 Anatomy tests

We have not been able to load the tests due to our OWL Parser.

15.5 Conclusions

15.5.1 General comments

In its latest version, OLA has proven a more robust tool for alignment than it was a year before. The results show a substantial progress has been made since the EON 2004 alignment contest. With respect to the performances of OLA at that forum, we made a big leap amounting to about 25% in both mean precision and mean recall.

Nevertheless, we see that a vast space for improvement lays ahead of our project. The weaknesses of the current similarity mechanisms can be summarized as follows. First, the tuning of the algorithm is still a rigid process. Indeed, while the weights can now be computed following a specific footprint of the ontology, a mechanism for the choice of a particular name similarity on the same basis has yet to be defined.

Second, although we take into account the biggest possible amount of knowledge about entities, there are sources of similarity that have been ignored so far, in particular entity comments.

15.5.2 Discussions on the way to improve the proposed system

Besides expanding the lexical processing to comments in entities and providing a flexible decision mechanism for the choice of the default name similarity, a possible improvement of the system will be the integration of a learning module for weight estimation. As for similarity, the biggest challenge here is to define the representation of the input data, i.e., the descriptors of the entries for the learning algorithm.

Another research track would be the definition of an optimal matching algorithm. In fact, the current procedures are sub-optimal in the sense that they only chose local optima for each aligned entity. Consequently, as strict 1:1 matchings are to be produced, a single bad choice could easily generate a chain of wrong alignment decisions and thus negatively impact the performances of the tool.

Part III

Semantics of matching and alignment

Chapter 16

Semantics for mappings

The previous chapter provided a very general view of what is to be found in mappings. However, it did not gave a precise semantics of the mapping that are expected from alignment and reconciliation processes. This chapter provides the semantics of these mappings.

The chapter begins with an example whose purpose is providing the reader with the intuition behind the introduced formalism. In the following sections syntax and semantics of the mappings are introduced. While in the last part of this chapter we briefly show that well known approaches to information integration fit into the described framework.

16.1 Motivating Example

Let's consider an example emphasising the difference between the rule-based semantics of integration and the classical semantics given to data integration systems, in the case of crisp mappings. Suppose we have three distributed information nodes. The first one (Σ_1) is the municipality's internal database, which has a binary table `Citizen-1` which contains the name of the citizen and the marital status (with values *single* or *married*). The second one (Σ_2) is a public database, obtained from the municipality's database, with two unary tables `Male-2` and `Female-2`. The third information node (Σ_3) is the Pension Agency database, obtained from a public database, with the unary table `Citizen-3` and a binary table `Marriage-3` (stating that two people are married). The three information nodes are interconnected by means of the following mappings:

$1 : \texttt{Citizen-1}(x, y) \leadsto_\alpha 2 : (\texttt{Male-2}(x) \vee \texttt{Female-2}(x))$
(this mapping connects Σ_1 with Σ_2)

$2 : \texttt{Male-2}(x) \leadsto_\alpha 3 : \texttt{Citizen-3}(x)$
$2 : \texttt{Female-2}(x) \leadsto_\alpha 3 : \texttt{Citizen-3}(x)$
(these mappings connect Σ_2 with Σ_3)

In the classical model, the `Citizen-3` table in Σ_3 should be filled with all of the individuals in the `Citizen-1` table in Σ_1, since the following mapping is logically implied:

$1 : \texttt{Citizen-1}(x) \leadsto_\alpha 3 : \texttt{Citizen-3}(x)$

However, in a rule-based integrated system – which can be compared to a peer-to-peer system – this is not a desirable conclusion. In fact, mappings should be interpreted only for fetching data, and not for deduction. In this example, the tables `Female-2` and `Male-2` in Σ_2 will be empty, since the data is fetched from Σ_1, where the gender of any specific entry in `Citizen-1` is not known. From the perspective of Σ_2, the only thing that is known is that each citizen is in the view ($\texttt{Female-2} \vee \texttt{Male-2}$). Therefore, when Σ_3 asks for data from Σ_2, the result will be empty. In other words, the mappings

$$2:\texttt{Male-2}(x) \rightsquigarrow_\alpha 3:\texttt{Citizen-3}(x)$$
$$2:\texttt{Female-2}(x) \rightsquigarrow_\alpha 3:\texttt{Citizen-3}(x)$$

will transfer no data from Σ_2 to Σ_3, since no individual is known in Σ_2 to be either definitely a male (in which case the first mapping would apply) or definitely a female (in which case the second mapping would apply). We only know that any citizen in Σ_1 is either male or female in Σ_2, and no reasoning about the mappings should be allowed.

Suppose now to have an additional cyclic pair of mappings connecting Σ_1 and Σ_3 as follows:

$$1:\texttt{Citizen-1}(x,\text{"married"}) \rightsquigarrow_\alpha 3:\texttt{Marriage-3}(x,y)$$
$$3:\texttt{Marriage-3}(x,y) \rightsquigarrow_\alpha 1:(\texttt{Citizen-1}(x,\text{"married"}) \wedge \texttt{Citizen-1}(y,\text{"married"}))$$

These cyclic mappings serve the purpose to *synchronise* the people who are known to be married from within the information node Σ_1 (by means of the `Citizen-1` table) with the people who are known to be married from within the information node Σ_3 (by means of the `Marriage-3` table).

Suppose that it is known in Σ_1 that only John is married, and nothing in known in Σ_3 about marriages. The cyclic mappings will propagate this information to Σ_3. However, there is still a subtle difference between the mappings interpreted in a classical way an the mappings interpreted as rules. In the classical model, a query to Σ_3 asking for the non existence of some married person different from John will get a negative answer. In a rule-based setting, we actually expect a positive answer, since the only information that is fetched is about John.

16.2 Syntax

Mappings are means to align knowledge among entities providing information. For this reason, the first concept to be introduced is a formal representation of these entities. These entities are called *information nodes*, and can be considered as first order theories on (possibly) distinct signatures.

However, the purpose of semantic alignment is to share knowledge among the nodes. Therefore, it is assumed a shared set of constant names which provides a sort of common vocabulary for the objects in the information system. One example of such shared constants are the URN in the world wide web.

Definition 40 (Information node). *Let I be a nonempty finite set of indexes $\{1, 2, \ldots, n\}$, and C be a set of constants. For each pair of distinct i, $j \in I$, let L_i be a first order function-free language with signature disjoint from L_j but for the shared constants C. An* information node Σ_i *is a theory on the first order language L_i.*

Given the starting blocks provided by the information nodes, the *mappings* are defined as relationships connecting formulae from different information nodes. As highlighted by the example, there are two different types of mappings according to the semantics of the integration among the nodes.

The mapping are distinguished into *classical* and *rule-based*. As their names suggest, their semantics differ in order to take into account the main two approaches in information integration. Only the syntax is described in this section, the formal semantics is introduced in the following section.

Definition 41 (Mapping). *A* mapping *is an expression of either of the form:*

$$\begin{array}{ll} i : \phi(\mathbf{x}) \Rightarrow_\alpha j : \psi(\mathbf{x}) & \textit{(classical mapping)} \\ i : \phi(\mathbf{x}) \rightsquigarrow_\alpha j : \psi(\mathbf{x}) & \textit{(rule-based mapping)} \end{array}$$

where i, j are distinct indices, $\alpha \in [\bot, \top]$ is a degree of confidence and ϕ is an open formula of L_i, and $\psi(\mathbf{x})$ is an open formula of L_j, both with free variables $\mathbf{x} = \{x_1, \ldots, x_\ell\}$. A crisp mapping *is a mapping with a degree of confidence $\alpha = \top$.*

In addition to generic formula mappings, a special kind of mappings is included to allows the alignment of arbitrary constants.

Definition 42 (Constant Mapping). *A* constant mapping *is an expression of the form $c_i \mapsto_\alpha c_j$ where i, j are distinct indices, $\alpha \in [\bot, \top]$ is a degree of confidence, c_i is a constant of Σ_i, c_j is a constant of Σ_j.*

To any mapping is associated a degree of confidence to allow the representation of uncertainty into the framework. This aspect is described in Section 16.4.

An integrated system is defined as the information nodes themselves, together with the mappings connecting them.

Definition 43 (Integrated system). *An* integrated system *is composed by a set MDB of information nodes and a set MAP of mappings.*

The purpose of an information system is to provide knowledge; therefore querying is an essential aspect of this framework. A query is always considered w.r.t. a given node, the alignment provides the mechanism in which different nodes can contribute to the answer of a given query. For this reason, queries are defined as formulae written with a language from an (arbitrary) single node.

Queries can have free variables, and in this case they retrieve set of tuples corresponding to the variables. When queries have no free variables, they are called boolean, since they can be either true or false.

Definition 44 (Query). *A* query *is a (possibly open) first order formula in the language of one of the information node Σ_i.*

16.3 Semantics of crisp mappings

In order to simplify the exposition in this section only crisp mappings are considered. The following section will take into account the degree of confidence as well.

To describe the semantics of the integrated system, the semantics of each node must be accounted for. Then the interconnection among the single nodes are considered, together with the restrictions imposed by the mappings.

It has been assumed that each node is a first order theory, therefore interpretations for each node are given in terms of a domain and first order interpretations. An interpretation for the integrated system consists in the set of interpretations for the single nodes. However, this is not sufficient because the node interpretation are not required to share the same domain.

Domains of different node interpretations are related by means of *match* relations, mapping elements of a domain to elements of the domain of a different node.

Definition 45 (Interpretation)**.** *Let $\langle MDB, MAP\rangle$ be an integrated system with crisp mappings only, and for each information node Σ_i let Δ_i be a non empty set of objects. For each pair of distinct indices in MDB, we define a* match *relation $R_{i,j} \subseteq \Delta_i \times \Delta_j$.*

An integrated interpretation *for the integrated system is a collection of information node models $\mathbf{m} = \{m_1, m_2, \ldots m_n\}$. For each information node Σ_i in MDB, an information node model m_i is a first order model of Σ_i on the domain Δ_i that interpret constants in C as themselves; i.e.,*

$$m_i \models \Sigma_i.$$

Models of an integration system are the interpretations which satisfy the mappings among the nodes.

Definition 46 (Models)**.** *Let's define an assignment α_i for the information node Σ_i in the usual way as a function from variable symbols in L_i to elements in Δ_i. In addition, we restrict the assignments to satisfy the match relations, i.e., $R_{i,j}(\alpha_i(x), \alpha_j(x))$ for each variable symbol x and each pair of distinct indices i, j in the integrated system.*

A model $\mathbf{M}$ *of an integrated system – written $\mathbf{M} \models \langle MDB, MAP\rangle$ – is a nonempty set of integrated interpretations satisfying every mapping, i.e, for each pair of assignments α_i and α_j the following holds:*

– *if the mapping is classical – $(i : \phi(\mathbf{x}) \Rightarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$\forall \mathbf{m} \in \mathbf{M}.\, ((\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x})) \rightarrow (\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x})))$$

– *if the mapping is rule-based – $(i : \phi(\mathbf{x}) \rightsquigarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$(\forall \mathbf{m} \in \mathbf{M}.(\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x}))) \rightarrow (\forall \mathbf{m} \in \mathbf{M}.(\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x})))$$

– *if the mapping is between constants – $c_i \mapsto_\alpha c_j$ – then*

$$\forall \mathbf{m} \in \mathbf{M}.\, ((\mathbf{m}|_i, \alpha_i \models (x = c_i)) \rightarrow (\mathbf{m}|_j, \alpha_j \models (x = c_j)))$$

where we intend $\mathbf{m}|_i$ to be the element m_i of $\mathbf{m}$.

Although the mappings are restricted to three kinds, the freedom in their combination allows to represent a variety of commonly used mappings. In fact, even the constant mapping can be represented by means of a classical mapping; as shown in the semantics above.

Among the widely used mappings, two common examples are equivalence and disjointness. The first one stating the equivalence between two formulae, and the second their disjointness. Given the nature of these two constraints they are better represented by means of classical mappings.

An equivalence mapping can be represented by means of two symmetric mappings. For example, to say that $Car(\cdot)$ in the node 1 is equivalent to $Voiture(\cdot)$ in node 2, the following two mappings can be used:

$$1 : Car(x) \Rightarrow_\alpha 2 : Voiture(x)$$
$$2 : Voiture(x) \Rightarrow_\alpha 1 : Car(x)$$

Disjointness mappings can be represented using negation in one of the formulae. For example, to say that two nodes, although they use the same predicate $Person(\cdot)$, contain informations about two different group of people the following mapping can be employed:

$$1 : Person(x) \Rightarrow_\alpha 2 : \neg Person(x)$$

Complex mappings can be represented using rule-based mappings as well; but the nature of their semantics makes their combination less intuitive.

Semantics for the queries is provided in the usual way, by means of the models of an integrated system. Note that answers to a query are given in terms of the shared constants.

Definition 47 (Query answer)**.** *Let $Q_i(\mathbf{x})$ be a query with free variables $\mathbf{x}$ (possibly empty). The* answer set *of Q_i is the set of substitutions of $\mathbf{x}$ with constants $\mathbf{c}$, such that any model $\mathbf{M}$ of an integrated system satisfies the query, i.e.,*

$$\{\mathbf{c} \in C \times \cdots \times C \mid \forall \mathbf{M}.\ (\mathbf{M} \models \langle MDB, MAP\rangle) \rightarrow \forall \mathbf{m} \in \mathbf{M}.\ (m_i \models Q_i(\mathbf{c}))\}$$

16.4 Semantics of fuzzy mappings

In real-life applications, the conceptualisation of the specific domain may result to a represented knowledge that has deficiencies. In general, the information represented can be imprecise, incomplete, vague, fragmentary, contradictory, random, etc. Moreover, the mappings between different information nodes are also caused by uncertainty.

Modelling this can take advantage of relationships between formulas which are not crisp (like $\Longrightarrow$), but have an α component which is different from $\top$ and $\bot$. We present here the semantics of the fuzzy one. In the framework presented here, we try to face this uncertainty, by using degrees (between 0 and 1) that represent the confidence of a specific hypothesis. Three types of uncertainty are introduced in the knowledge representation and alignment process:

– Fuzzy interpretations, i.e. a degree of membership to each interpretation (different semantics for the constructors of the representation language).
– Fuzzy mappings, i.e. a degree of confidence associated to each mapping.
– Fuzzy alignment, i.e. a degree of trust associated to the alignment system.

In this section, we concentrate on the first and the second types of uncertainty. We assume that we have mappings (crisp or not) between information nodes constructed with the aid of a fuzzy extension of its representation language. This means that the syntactic constructors of the language have different semantics based on the notion of a fuzzy interpretation. It is important to notice that all the above constructors should satisfy some minimal requirements that ensure the validity of the extension. In Deliverable 2.5.1 (Specification of Coordination of Rule and Ontology Languages), and more specifically in Section 5 (A Fuzzy Extension), a fuzzy DL extension is presented and the above requirements are summarised in the definition of *valid fuzzy assertional extensions*. The use of the extended language L_i (with extended semantics) results to formulas that have truth values between 0 and 1 (and not only 0 or 1), under a fuzzy model m_i of Σ_i (on the domain Δ_i) and an assignment α.

Definition 48 (Semantics of fuzzy mappings)**.** *Let $\langle MDB, MAP \rangle$ be an integrated system with information nodes and mappings that are crisp or not. Let also R be a fuzzy match relation $R_{i,j} : \Delta_i \times \Delta_j \to [0,1]$.*
An integrated interpretation *for the integrated system is a collection of fuzzy models $\mathbf{m} = \{m_1, m_2, \dots m_n\}$, where m_i is a fuzzy model of Σ_i.*
A model $\mathbf{M}$ *of an integrated system is a nonempty set of integrated interpretations satisfying every mapping, i.e, for each pair of assignments α_i and α_j that satisfy $R_{i,j}(\alpha_i(x), \alpha_j(x))$ (i.e., $R_{i,j}(\alpha_i(x), \alpha_j(x)) > 0$) the following holds:*

– *if the mapping is classical – $(i : \phi(\mathbf{x}) \Rightarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$\inf_{\mathbf{m} \in \mathbf{M}} \omega_t((\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x})), (\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x}))) \geqslant \alpha$$

– *if the mapping is rule-based – $(i : \phi(\mathbf{x}) \rightsquigarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$\omega_t[\inf_{\mathbf{m} \in \mathbf{M}} (\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x})), \inf_{\mathbf{m} \in \mathbf{M}} (\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x}))] \geqslant \alpha$$

where ω_t is a fuzzy implication, t is a triangular norm.

16.5 Comparison with other approaches

Classical logic-based Information Integration If we consider an integrated system where there is a unique common domain Δ for each information node, the match relation is the identity relation over Δ, and only classical mappings are present, then the logical framework exactly characterises (and generalises) the classical logic-based information integration approach [Franconi *et al.*, 2001; Catarci and Lenzerini, 1993; Calvanese *et al.*, 1998b; Jarke *et al.*, 1999; Jarke *et al.*, 2000; Calvanese *et al.*, 2002a; Peim *et al.*, 2004].

Consider, as an example, the case of multiple databases to be integrated. Each database have its own conceptual schema and logical schema, where the logical schema can be seen a set of views over the conceptual schema (local-as-view approach). We assume that each symbol of each schema is identified by a unique global symbol; i.e., the various databases have disjoint signatures. Interdependencies between entities and relationships in different schemas are represented by means of integrity constraints involving symbols of the schemas. Such interdependencies are called *inter-model assertions*. The union of the various schemas with the inter-model assertions and the local views forms the global integrated schema, or the *mediator*. It is worth noting that the integration process is incremental – since the integrated schema can be monotonically refined as soon as there is new understanding of the different component schemas – and that the resulting unified schema is strongly dependent from (actually, it includes) the schemas of the single information sources.

This approach gives both a clear semantics to the integration process of ontologies, and a calculus for deriving inconsistencies and checking the validity of integrity constraints in the integrated schema. Most importantly, in this framework global queries can be defined as views over single ontologies, or they can be generalised to span over multiple ontologies. The view-based query processing mechanism will guarantee the correct answer to the global query from the local sources [Calì *et al.*, 2004].

In [Lenzerini, 2002] a comparison is given between the above local-as-view approach to processing global queries and the global-as-view approach, which is more common in current information integration architectures.

Only recently has knowledge representation research started to have an interest in query processing and information access. Recent work has come up with advanced reasoning techniques for query do2002a and rewriting using views under the constraints given by the ontology – also called view-based query processing [Ullman, 1997; Calvanese *et al.*, 2000b]. This means that the notion of accessing information through the navigation of an Ontology modelling the document's domain – which can be seen as a conceptual schema – has its formal foundations.

Two approaches to view-based query processing exist, namely query rewriting (see, e.g., [Beeri *et al.*, 1997]) and query answering (see, e.g., [Abiteboul and Duschka, 1998; Calvanese *et al.*, 2000a; Peim *et al.*, 2002]). In the former approach, we are given a query Q, a set of view definitions characterising the actual data, and a set of (conceptual) constraints – all over the conceptual vocabulary – and the goal is to reformulate the query into an expression, the rewriting, that refers only to the views, and provides the answer to Q. Typically, the rewriting is formulated in the same language used for the query and the views. In the latter approach, besides Q, the view definitions and the constraints, we are also given the extensions of the (materialised) views. The goal is to compute the set of tuples that are implied by these extensions, i.e., the set of tuples that are in the answer set of Q in all the databases that are consistent with the views and the constraints.

In both cases, view definitions can be characterised in the framework presented in this document. In fact, the mappings are general enough to be used to define queries over the different databases. Analysing the techniques for answering these queries is outside the scope of this document; however, with opportune restrictions the techniques presented in literature can be used in this framework.

Rule-based Information Integration If we consider an integrated system where there is a unique common domain Δ for each information node, the match relation is the identity relation over Δ, and only rule-based mappings are present, then the logical framework exactly characterises (and generalises) the peer-to-peer logic-based information integration approach. If we push further, by allowing arbitrary distinct domains for the information nodes as well as a general match relation, then the logical framework characterises the context based approach. In the following, we will briefly show how the most relevant approaches in the literature are actually within our proposed logical framework.

The autoepistemic approach, which is the basis for the rule-based semantics, was first introduced by [Donini *et al.*, 1998], with the goal of formalising the *constraint rules* implemented in many practical knowledge representation systems. These rules are also the basis of the recent formalisations of peer-to-peer systems [Franconi *et al.*, 2003]. As shown in [Franconi *et al.*, 2003], the autoepistemic semantics as defined above is equivalent to the context-based semantics of [Giunchiglia, 1993; Ghidini and Serafini, 1998; Ghidini and Giunchiglia, 2001], and to the use of the autoepistemic operator, as defined, e.g., in [Reiter, 1992].

The framework presented in this document shares the same spirit of the Piazza system [Halevy *et al.*, 2003; Tatarinov and Halevy, 2004]. The vision of the Piazza peer data management system (PDMS) project is to provide semantic mediation between an environment of peers, each with its own schema. Rather than requiring the use of a single, uniform, centralised mediated schema to share data between peers, Piazza allows peers to define semantic mappings between pairs of peers (or among small subsets of peers). In turn, transitive relationships among the schemas of the peers are exploited so the entire resources of the PDMS can be used. The Piazza system is limited in the fact that it does not allow full GLAV mapping rules (i.e., heads must be atomic queries), it does not allow for cyclic mapping rules, and it does not allow for dynamic networks.

In the field of PDMS as defined above – which includes – there are only two other approaches which deal in a well founded way with cycles in the mapping rules [Serafini and Ghidini, 2000; Calvanese *et al.*, 2003]. The acyclic case is relatively simple – a query is propagated through the network until it reaches the leaves of the network. The work in [Calvanese *et al.*, 2003] uses a notion of semantics similar to the semantics introduced in [Franconi *et al.*, 2003], but it describes a partially distributed algorithm, that assumes that nodes may exchange mappings and data, so that a unique node will eventually evaluate in one shot the query answer – there is no distributed computation and the network may be flooded with data. The paper [Serafini and Ghidini, 2000] describes a local algorithm to compute query answers, but it does not allow real GLAV mapping rules (with existential variables in the head).

Chapter 17

A formal investigation of mapping languages for terminological knowledge

The benefits of using ontologies as explicit models of the conceptualization underlying information sources has widely been recognized. Meanwhile, a number of logical languages for representing and reasoning about ontologies have been proposed and there are even language standards now that guarantee stability and homogeneity on the language level. At the same time, the need to represent ontology alignment by means of mappings between different ontologies has been recognized as a result of the fact that different ontologies may partially overlap or even represent the same domain from different points of view [Bouquet *et al.*, 2004b]. As a result, a number of proposals have been made for extending ontology languages with notions of mappings between different models. Unlike for the case of ontology languages, work on languages to represent ontology mappings has not yet reached a state where a common understanding of the basic principles exists. As a consequence, existing proposals show major differences concerning almost all possible aspects of such languages. This makes it difficult to compare approaches and to make a decision about the usefulness of a particular approach in a given situation.

The purpose of this work is to provide a better understanding of the commonalities and differences of existing proposals for ontology mapping languages. We restrict our attention to logic-based approaches that have been defined as extensions of existing formalisms for representing Terminological Knowledge. In particular, we chose approaches that extend description logics (DL) with notions of mappings between different T-boxes. The rationale for this choice is the fact that DLs are a widely agreed standard for describing terminological knowledge. In particular, DLs have gained a lot of attention as a standardized way of representing ontologies on the Semantic Web [Horrocks *et al.*, 2003].

Approach and Contributions

We encode the different mapping languages in an extended version of distributed first-order logic (DFOL), a logical framework for representing distributed knowledge systems [Ghidini and Serafini, 1998]. DFOL consists of two components: a family of first order theories and a set of axioms describing the relations between these theories. As most proposals for mapping languages are based on a subset of first-order logic for describing local models and mappings with a particular semantics for the connections between models, these mapping languages can be expressed in distributed first order logic in the following way:

– restrictions on the use of first order sentences for describing domain models
– the form of axioms that can be used for describing relations between domain models
– axioms describing the assumptions that are encoded in the specific semantics of mappings

Encoding the different mapping approaches in first-order logic in this way has several advantages for an analysis and comparison of existing work. In particular it allows us to do a formal analysis and comparison of different approaches in a uniform logical framework. In the course of the investigations, we make the following contributions to the state of the art in distributed knowledge representation and reasoning:

– we show how the DFOL formalism can be used to model relations between heterogeneous domains
– we encode existing mapping approaches in a common framework, making them more comparable
– we make hidden assumptions explicit in terms of distributed first order logic axioms
– we provide first results on the relative expressiveness of the approaches and identify shared fragments

The chapter is structured as follows. In section 17.1 we introduce distributed first order logic as a general model for describing distributed knowledge systems. We explain the intuition of the logic and introduce its syntax and semantics. In section 17.2 we describe how the different mapping approaches can be encoded in distributed first order language. Here we will focus on the representation of mappings and the encoding of hidden assumptions. In section 17.3 we compare the different approaches based on their encoding in DFOL and discuss issues such as relative expressiveness and compatibility of the different approaches and conclude with a summary of our findings and open questions.

17.1 Distributed First-Order Logic

This section introduces distributed first order logic as a basis for modeling distributed knowledge bases. More details about the language including a sound and complete calculus can be found in [Ghidini and Serafini, 1998].

Let $\{L_i\}_{i \in I}$ (in the following $\{L_i\}$) be a family of first order languages with equality defined over a non-empty set I of indexes. Each language L_i is the language used by the i-th knowledge base (ontology). $\{L_i\}$ may intersect — but do not need to intersect. The signature of L_i is extended with a new set of symbols used to denote objects which are related with other objects in different ontologies. For each variable x, and each index $j \in I$ with $j \neq i$ we have two new symbols $x^{\rightarrow j}$ and $x^{j \rightarrow}$, called *arrow variables*. Terms and formulas of L_i, also called *i-terms* and *i-formulas* are defined in the usual way. Quantification on arrow variables is not permitted. The notation $\phi(\mathbf{x})$ is used to denote the formula ϕ and the fact that the free variables of ϕ are $\mathbf{x} = \{x_1, \ldots, x_n\}$. In order to distinguish occurrences of terms and formulas in different languages we label them with their index. The expression $i : \phi$ denotes the formula ϕ of the i-th knowledge base.

The semantics of DFOL is an extension of Local Models Semantics defined in [Ghidini and Giunchiglia, 2001]. Local models are defined in terms of first order models. To capture the fact that certain predicates are completely known by the i-th sub-system we select a sub-language of L_i containing the equality predicate, denoted as L_i^c, which we call the *complete fragment* of L_i. *Complete terms* and *complete formulas* are terms and formulas of L_i^c and vice versa.

Definition 49 (Set of local Models)**.** *A* set of local models *of L_i is a set of first order interpretations of L_i, on a domain $\mathbf{dom}_i$, which agree on the interpretation of L_i^c, the complete fragment of L_i.*

As noted in [Franconi and Tessaris, 2004] there is a foundational difference between approaches that use epistemic states and approaches that use a classical model theoretic semantics. The two approaches differ as long as there is more than one model m. Using the notion of complete sublanguage L_c, however, we can force that the set of local models is either a singleton or the empty set by enforcing that $L^c = L$. Under this assumption the two ways of defining the semantics are equivalent. Using this assumption, we are therefore able to simulate both kinds of semantics in DFOL.

Two or more models can carry information about the same portion of the world. In this case we say that they *semantically overlap*. Overlapping is unrelated to the fact that the same constant appears in two languages, as from the local semantics we have that the interpretation of a constant c in L_i is independent from the interpretation of the very same constant in L_j, with $i \neq j$. Overlapping is also unrelated to the intersection between the interpretation domains of two or more contexts. Namely if $\mathbf{dom}_1 \cap \mathbf{dom}_2 \neq \emptyset$, it does not mean that L_1 and L_2 overlap. Instead, DFOL explicitly represents semantic overlapping via a domain relation.

Definition 50 (Domain relation)**.** *A* domain relation *from $\mathbf{dom}_i$ to $\mathbf{dom}_j$ is a binary relation $r_{ij} \subseteq \mathbf{dom}_i \times \mathbf{dom}_j$.*

A domain relation from i to j represents the capability of the j-th sub-system to represent in its domain the domain of the i-th subsystem. A pair $\langle d, d' \rangle$ being in r_{ij} means that, from the point of view of j, d in $\mathbf{dom}_i$ is the representation of d' in $\mathbf{dom}_j$. We use the functional notation $r_{ij}(d)$ to denote the set $\{d' \in \mathbf{dom}_j | \langle d, d' \rangle \in r_{ij}\}$. The domain relation r_{ij} formalizes j's subjective point of view on the relation between $\mathbf{dom}_i$ and $\mathbf{dom}_j$ and not an absolute objective point of view. Or in other words $r_{ij} \neq r_{ji}$ because of the non-symmetrical nature of mappings. Therefore $\langle d, d' \rangle \in r_{ij}$ must not be read as if d and d' were the same object in a domain shared by i and j. This fact would indeed be formalized by some observer which is external (above, meta) to both i and j who will state that d and d' corresponds to the same real world object. Using the notion of domain relation, we can define the notion of a DFOL model for a set of local models.

Definition 51 (DFOL Model)**.** *A DFOL* model *$\mathcal{M}$ is a pair $\langle \{\mathcal{M}_i\}, \{r_{ij}\} \rangle$ where, for each $i \neq j \in I$: $\mathcal{M}_i$ is a set of local models for L_i, and r_{ij} is a domain relation from $\mathbf{dom}_i$ to $\mathbf{dom}_j$.*

In the following we will sometimes need to specify the set of tuples of objects that belongs to the interpretation of a predicate P in all the local models in S_i. We use the notation $\|P(\mathbf{x})\|_i$ (if it is not necessary we can omit the variables, using the simpler notation $\|P\|_i$) to indicate such a set. Formally $\|P(\mathbf{x})\|_i = \bigcap_{m=\langle \mathbf{dom}, \mathcal{I} \rangle \in S_i} P^{\mathcal{I}}$. $\|P\|_i$ intuitively indicates the set of objects that are known to be P by the sub-system i.

We extend the classical notion of assignment (e.g., the one given for first order logic) to deal with arrow variables using domain relations. In particular, an assignment a, provides for each system i, an interpretation for all the variables, and for *some* (but not necessarily all) arrow variables, as the domain relations might be such that there is no consistent way to assign arrow variables. For instance if $a_i(x) = d$ and $r_{ij}(d) = \emptyset$, then a_j cannot assign anything to $x^{i\rightarrow}$.

Definition 52 (Assignment)**.** *Let $\mathcal{M} = \langle \{\mathcal{M}_i\}, \{r_{ij}\} \rangle$ be a model for $\{L_i\}$. An* assignment *a is a family $\{a_i\}$ of partial functions from the set of variables and arrow variables to $\mathbf{dom}_i$, such that for each variable x and all $i \neq j$:*

1. $a_i(x) \in \mathbf{dom}_i$*;*
2. $a_i(x^{j\rightarrow}) \in r_{ji}(a_j(x))$*;*
3. $a_j(x) \in r_{ij}(a_i(x^{\rightarrow j}))$*;*

An assignment a *is* admissible for a formula $i\!:\!\phi$ *if* a_i *assigns all the arrow variables occurring in* ϕ*. Furthermore,* a *is admissible for a set of formulas* Γ *if it is admissible for any* $j : \phi \in \Gamma$*. An assignment* a *is* strictly admissible *for a set of formulas* Γ *if it is admissible for* Γ *and assigns only the arrow variables that occur in* Γ*.*

The intuition on how arrow variables are assigned is the following. If the variable x occurring in $i : \phi$ is thought as a placeholder for a generic element $d \in \mathbf{dom}_i$, the arrow variable $x^{\rightarrow i}$ occurring in $j : \psi$ is a placeholder for an element $d' \in \mathbf{dom}_j$ which is a pre-image (via r_{ji}) of d. Analogously the extended variable $x^{i\rightarrow}$ occurring in $k : \psi$ is a placeholder for any element $d'' \in \mathbf{dom}_k$ which is an image (via r_{ik}) of d. This situation is illustrated in the following drawing:

$$\begin{tikzcd}
\text{Languages} & L_j & L_i & L_k \\
mboxSymbols & x^{\rightarrow i} \arrow[d, "a_j"] & x \arrow[d, "a_i"] & x^{i\rightarrow} \arrow[d, "a_k"] \\
\text{Semantics} & d' \arrow[r, "r_{ji}"] & d \arrow[r, "r_{ik}"] & d''
\end{tikzcd}$$

Using the notion of an admissible assignment given above, satisfiability in distributed first order logic is defined as follows:

Definition 53 (Satisfiability)**.** *Let* $\mathcal{M} = \langle\{\mathcal{M}_i\}, \{r_{ij}\}\rangle$ *be a model for* $\{L_i\}$*,* $m \in \mathcal{M}_i$*, and* a *an assignment. An* i*-formula* ϕ *is* satisfied *by* m*, w.r.t,* a*, in symbols* $m \models_D \phi[a]$ *if*

1. a *is admissible for* $i\!:\!\phi$ *and*
2. $m \models \phi[a_i]$*, according to the definition of satisfiability for first order logic.*

$\mathcal{M} \models \Gamma[a]$ *if for all* $i\!:\!\phi \in \Gamma$ *and* $m \in \mathcal{M}_i$*,* $m \models_D \phi[a_i]$[1].

Mappings between different knowledge bases are formalized in DFOL by a new form of constraints that involves more than one knowledge base. These formulas that will be the basis for describing different mapping approaches are called interpretation constraints and are defined as follows:

Definition 54 (Interpretation constraint)**.** *An* interpretation constraint from $i_1, \ldots, i_n$ to i *with* $i_k \neq i$ *for* $1 \leq k \leq n$ *is an expression of the form*

$$i_1\!:\!\phi_1, \ldots, i_n\!:\!\phi_n \rightarrow i\!:\!\phi \tag{17.1}$$

The interpretation constraint (17.1) can be considered as an axiom that restricts the set of possible DFOL models to those which satisfy it. Therefore we need to define when a DFOL model satisfies an interpretation constraint.

[1]Since it will be clear from the context, in the remainder we will use the classical satisfiability symbol $\models$ instead of $\models_D$ and we will write $m \models \phi[a]$ to mean that an i-formula ϕ is satisfied by m. In writing $m \models \phi[a]$ we always mean that a is admissible for $i\!:\!\phi$ (in addition to the fact that m classically satisfies ϕ under the assignment a).

a)	$\mathcal{M} \models i\!:\!P(x^{\rightarrow j}) \rightarrow j\!:\!Q(x)$	iff	For all $d \in \|P\|_i$ and $\underline{\text{for all } d' \in r_{ij}(d)}$, $d' \in \|Q\|_j$
b)	$\mathcal{M} \models i\!:\!P(x) \rightarrow j\!:\!Q(x^{i\rightarrow})$	iff	For all $d \in \|P\|_i$ $\underline{\text{there is a } d' \in r_{ij}(d)}$, s.t., $d' \in \|Q\|_j$
c)	$\mathcal{M} \models j\!:\!Q(x^{i\rightarrow}) \rightarrow i\!:\!P(x)$	iff	For all $d \in \|Q\|_j$ and $\underline{\text{for all } d'}$ $\underline{\text{with } d \in r_{ij}(d')}$, $d' \in \|P\|_i$
d)	$\mathcal{M} \models j\!:\!Q(x) \rightarrow i\!:\!P(x^{\rightarrow j})$	iff	For all $d \in \|Q\|_i$ $\underline{\text{there is a } d'}$ $\underline{\text{with } d \in r_{ij}(d')}$, s.t., $d' \in \|P\|_i$

Figure 17.1: Implicit Quantification of Arrow Variables in Interpretation Constraints

Definition 55 (Satisfiability of interpretation constraints)**.** *A model $\mathcal{M}$* satisfies *the interpretation constraint (17.1), in symbols $\mathcal{M} \models i_1 : \phi_1, \ldots, i_n : \phi_n \rightarrow i : \phi$ if for any assignment a strictly admissible for $\{i_1\!:\!\phi_1, \ldots, i_n\!:\!\phi_n\}$, if $\mathcal{M} \models i_k\!:\!\phi_k[a]$ for $1 \leq k \leq n$, then a can be extended to an assignment a' admissible for $i\!:\!\phi$ and such that $\mathcal{M} \models i\!:\!\phi[a']$.*

Notice that, depending on whether an arrow variable $x^{\rightarrow}$ occurs on the left or on the right side of the constraint, $x^{\rightarrow}$ has a universal or an existential reading. Figure 17.1 summarizes the different possible readings that will reoccur later. Notationally, for any predicate P, $\|P\|_i = \bigcap_{m \in \mathcal{M}_i} m(P)$, where $m(P)$ is the interpretation of P in m.

We like to explain these four kind of interpretation constraints in terms of the two ontologies of Figure 18.2. The first ontology $\mathbf{O_1}$ is designed by an expert of vehicles but still have some knowledge about houses. The second ontology $\mathbf{O_2}$ is the opposite: it has detailed knowledge about house but only fundamental knowledge about vehicles. In a set of mappings we may want to express for example that a $Volant$ which is copied to $\mathbf{O_2}$ is an $Vehicle$. It can be expressed by the following interpretation constraint a): $1\!:\!Volant(x^{\rightarrow 2}) \rightarrow 2\!:\!Vehicle(x)$. An instantiation of interpretation constraint b) is that every $Maison$ in ontology $\mathbf{O_1}$ has to be copied to $\mathbf{O_2}$ and is an $Domicile$: $1\!:\!Maison(x) \rightarrow 2\!:\!Domicile(x^{1\rightarrow})$. Also the opposite direction can be expressed. Every $seaplane$ which is copied to $\mathbf{O_1}$ has to be a $Hydravion$. In an interpretation constraint c) it can be expressed as $2 : Seaplane(x^{1\rightarrow}) \rightarrow 1 : Hydravion(x)$. The last type of interpretation constraint d) express with $2\!:\!Vehicle(x) \rightarrow 1\!:\!Vehicule(x^{\rightarrow 2})$ that every $Vehicle$ is a $Vehicule$.

By means of interpretation constraints on equality, we can formalize possible relations between heterogeneous domains.

$$
\begin{array}{rcl}
\mathsf{F}_{ij} & = & \{i\!:\!x^{\rightarrow j} = y^{\rightarrow j} \rightarrow j\!:\!x = y\} \\
\mathsf{INV}_{ij} & = & \left\{ \begin{array}{l} i\!:\!x = y^{j\rightarrow} \rightarrow j\!:\!x^{i\rightarrow} = y \\ j\!:\!x = y^{i\rightarrow} \rightarrow i\!:\!x^{j\rightarrow} = y \end{array} \right\} \\
\mathsf{OD}_{ij} & = & \mathsf{F}_{ij} \cup \mathsf{F}_{ji} \cup \mathsf{INV}_{ij} \\
\mathsf{ED}_{ij} & = & \mathsf{OD}_{ij} \cup \{i\!:\!x = x \rightarrow j\!:\!x^{i\rightarrow} = x^{i\rightarrow}\} \\
\mathsf{ID}_{ij} & = & \mathsf{ED}_{ij} \cup \mathsf{ED}_{ji} \\
\mathsf{RD}_{ij} & = & \left\{ \begin{array}{l} i\!:\!x = c \rightarrow j\!:\!x^{i\rightarrow} = c \\ j\!:\!x = c \rightarrow i\!:\!x^{j\rightarrow} = c \end{array} \middle| \, c \in L_i \cap L_j \right\} \\
\mathsf{IP}_{ij} & = & i\!:\!\bot \rightarrow j\!:\!\bot
\end{array}
$$

Proposition 1. *Let $\mathcal{M}$ be a DFOL model and $i \neq j \in I$.*

1. *$\mathcal{M} \models \mathsf{F}_{ij}$ iff r_{ij} is a partial function.*
2. *$\mathcal{M} \models \mathsf{INV}_{ij}$ iff r_{ij} is the inverse of r_{ji}.*
3. *$\mathcal{M} \models \mathsf{OD}_{ij}$ if $r_{ij}(= r_{ji}^{-1})$ is an isomorphism between a subset of $\mathbf{dom}_i$ and a subset of $\mathbf{dom}_j$. I.e., $\mathbf{dom}_i$ and $\mathbf{dom}_j$ overlap.*

4. *$\mathcal{M} \models \mathsf{ED}_{ij}$ iff $r_{ij}(= r_{ji}^{-1})$ is an isomorphism between $\mathbf{dom}_i$ and a subset of $\mathbf{dom}_j$. I.e., $\mathbf{dom}_i$ is (isomorphically) embedded in $\mathbf{dom}_j$*
5. *$\mathcal{M} \models \mathsf{ID}_{ij}$ iff $r_{ij}(= r_{ji}^{-1})$ is an isomorphism between $\mathbf{dom}_i$ and $\mathbf{dom}_j$. I.e., $\mathbf{dom}_i$ is isomorphic to $\mathbf{dom}_j$.*
6. *$\mathcal{M} \models \mathsf{RD}$ implies for every constant c of L_i and L_j, if c is interpreted in d for all $m \in \mathcal{M}_i$ then c is interpreted in $r_{ij}(d)$ for all models of $m \in \mathcal{M}_j$, and vice-versa. I.e., the constant c is rigidly interpreted by i and j in (two) corresponding objects.*
7. *Finally $\mathcal{M} \models \mathsf{IP}_{ij}$ iff $\mathcal{M}_i = \emptyset$ implies that $\mathcal{M}_j = \emptyset$. I.e., inconsistency propagates from i to j.*

17.2 Modeling Mapping Languages in DFOL

Formalisms for mapping languages are based on four main parameters: local languages and local semantics used to specify the local knowledge, and mapping languages and semantics for mappings, used to specify the semantic relations between the local knowledge. In this section we focus on the second pairs and as far as local languages and local semantics it is enough to notice that

Local languages In all approaches local knowledge is expressed by a suitable fragment of first order languages.

Local semantics with the notable exception of [Franconi and Tessaris, 2004], where authors propose an *epistemic approach* to information integration (see chapter 5 in deliverable D2.2.1 "Specification of a common framework for characterizing alignment"), all the other formalisms for ontology mapping assume that each local knowledge is interpreted in a (partial) state of the world and not into an epistemic state. This formally corresponds to the fact that each local knowledge base is associated with *at most one* FOL interpretation. The case of incomplete local knowledge will be described in the future.

The first assumption is naturally captured in DFOL, by simply considering L_i to be an adequately restricted FOL language. Concerning the local semantics, in DFOL models each L_i is associated with a *set of interpretations*. To simulate the single local model assumption, in DFOL it is enough to declare each L_i to be a *complete* language. This implies that all the $m \in M_i$ have to agree on the interpretation of L_i-symbols.

Notationally, $\phi, \psi, \ldots$ will be used to denote both DL expressions and FOL open formulas. If ϕ is a DL concept, $\phi(x)$ (or $\phi(x_1, \ldots, x_n)$) will denote the corresponding translation of ϕ in FOL as described in [Borgida, 1996]. If ϕ is a role R then $\phi(x, y)$ denotes its translation $R(x, y)$, and if ϕ is a constant c, then $\phi(x)$ denote its translation $x = c$. Finally we use $\mathbf{x}$ to denote a vector $x_1, \ldots, x_n$ of variables.

17.2.1 Distributed Description Logics/C-OWL

The approach presented in [Borgida and Serafini, 2003b] extends DL with a local model semantics similar to the one introduced above and so-called bridge rules to define semantic relations between different T-Boxes. A distributed interpretation for DDL on a family of DL languages $\{L_i\}$, is a family $\{\mathcal{I}_i\}$ of interpretations, one for each L_i plus a family $\{r_{ij}\}_{i \neq j \in I}$ of domain relations. While the original proposal only considered subsumption between concept expressions, the model was extended to a set of five semantic relations discussed below. The semantics of the five semantic relations defined in C-OWL is the following:

Definition 56 ([Bouquet *et al.*, 2004b]). *Let ϕ and ψ be either concepts, or individuals, or roles of the descriptive languages L_i and L_j respectively*[2].

1. $\mathfrak{I} \models i:\phi \xrightarrow{\sqsubseteq} j:\psi$ *if* $r_{ij}(\phi^{\mathcal{I}_i}) \subseteq \psi^{\mathcal{I}_j}$;
2. $\mathfrak{I} \models i:\phi \xrightarrow{\sqsupseteq} j:\psi$ *if* $r_{ij}(\phi^{\mathcal{I}_i}) \supseteq \psi^{\mathcal{I}_j}$;
3. $\mathfrak{I} \models i:\phi \xrightarrow{\equiv} j:\psi$ *if* $r_{ij}(\phi^{\mathcal{I}_i}) = \psi^{\mathcal{I}_j}$;
4. $\mathfrak{I} \models i:\phi \xrightarrow{\perp} j:\psi$ *if* $r_{ij}(\phi^{\mathcal{I}_i}) \cap \psi^{\mathcal{I}_j} = \emptyset$;
5. $\mathfrak{I} \models i:\phi \xrightarrow{*} j:\psi$ *if* $r_{ij}(\phi^{\mathcal{I}_i}) \cap \psi^{\mathcal{I}_j} \neq \emptyset$;

An interpretation for a context space is a model for it if all the bridge rules are satisfied.

From the above satisfiability condition one can see that the mapping $i:\phi \xrightarrow{\equiv} j:\psi$ is equivalent to the conjunction of the mappings $i:\phi \xrightarrow{\sqsubseteq} j:\psi$ and $i:\phi \xrightarrow{\sqsupseteq} j:\psi$. The mapping $i:\phi \xrightarrow{\perp} j:\psi$ is equivalent to $i : \phi \xrightarrow{\sqsubseteq} j : \neg\psi$. And finally the mapping $i : \phi \xrightarrow{*} j : \psi$ is the negation of the mapping $i:\phi \xrightarrow{\perp} j:\psi$. As the underlying notion of a model is the same as for DFOL, we can directly try to translate bridge rules into interpretation constraints. In particular, there are no additional assumptions about the nature of the domains that have to be modeled. The translation is the following:

C-OWL	**DFOL**
$i:\phi \xrightarrow{\sqsubseteq} j:\psi$	$i:\phi(x^{\rightarrow j}) \rightarrow j:\psi(x)$
$i:\phi \xrightarrow{\sqsupseteq} j:\psi$	$j:\psi(x) \rightarrow i:\phi(x^{\rightarrow j})$
$i:\phi \xrightarrow{\equiv} j:\psi$	$i:\phi(x^{\rightarrow j}) \rightarrow j:\psi(x)$ and $i:\phi(x^{\rightarrow j}) \rightarrow j:\psi(x)$
$i:\phi \xrightarrow{\perp} j:\psi$	$i:\phi(x^{\rightarrow j}) \rightarrow j:\neg\psi(x)$
$i:\phi \xrightarrow{*} j:\psi$	No translation

We see that a bridge rule basically corresponds to the interpretation a) and d) in Figure 17.1. The different semantic relations correspond to the usual readings of implications. Finally negative information about mappings (i.e., $i : \phi \overset{\sqsubseteq}{\not\longrightarrow} j : \psi$ is not representable by means of DFOL interpretation constraints.

17.2.2 Ontology Integration Framework (calvanese2002a)

Calvanese and colleagues in [Calvanese *et al.*, 2002b] propose a framework for mappings between ontologies that generalizes existing work on view-based schema integration [Ullman, 1997] and subsumes other approaches on connecting DL models with rules. In particular, they distinguish global centric, local centric and the combined approach. These approaches differ in the types of expressions connected by mappings. With respect to the semantics of mappings, they do not differ and we therefore treat them as one.

calvanese2002a assumes the existence of a global ontology g into which all local models s are mapped. On the semantic level, the domains of the local models are assumed to be embedded in

[2]In this definition, to be more homogeneous, we consider the interpretations of individuals to be sets containing a single object rather than the object itself.

a global domain. Further, in calvanese2002a constants are assumed to rigidly designate the same objects across domain. Finally, global inconsistency is assumed, in the sense that the inconsistency of a local knowledge base makes the whole system inconsistent. As shown in Proposition 1, we can capture these assumptions by the set of interpretation constraints ED_{sg}, RD_{sg}, and IP_{sg}, where s is the index of any source ontology and g the index of the global ontology.

According to these assumptions mappings are described in terms of correspondences between a local and the global model. The interpretation of these correspondences are defined as follows:

Definition 57 ([Calvanese *et al.*, 2002b]). *Correspondences between source ontologies (with interpretation $\mathcal{D}$) and global ontology (with interpretation $\mathcal{I}$) are of the following four forms*

1. *$\mathcal{I}$ satisfies $\langle \phi, \psi, sound\rangle$ w.r.t. the local interpretation $\mathcal{D}$, if all the tuples satisfying ψ in $\mathcal{D}$ satisfy ϕ in $\mathcal{I}$*
2. *$\langle \phi, \psi, complete\rangle$ w.r.t. the local interpretation $\mathcal{D}$, if no tuple other than those satisfying ψ in $\mathcal{D}$ satisfies ϕ in $\mathcal{I}$,*
3. *$\langle \phi, \psi, exact\rangle$ w.r.t. the local interpretation $\mathcal{D}$, if the set of tuples that satisfies ψ in $\mathcal{D}$ is exactly the set of tuples satisfying ϕ in $\mathcal{I}$.*

From the above semantic conditions, $\langle \phi, \psi, exact\rangle$ is equivalent to the conjunction of $\langle \phi, \psi, sound\rangle$ and $\langle \phi, \psi, complete\rangle$. It's therefore enough to provide the translation of the first two correspondences. The definitions 1 and 2 above can directly be expressed into interpretation constraints (compare Figure 17.1) resulting in the following translation:

GLAV Correspondence	DFOL
$\langle \phi, \psi, sound\rangle$	$s:\psi(\mathbf{x}) \to g:\phi(\mathbf{x}^{s\to})$
$\langle \phi, \psi, complete\rangle$	$g:\phi(\mathbf{x}) \to s:\psi(\mathbf{x}^{\to g})$
$\langle \phi, \psi, exact\rangle$	$s:\psi(\mathbf{x}) \to g:\phi(\mathbf{x}^{s\to})$ and $g:\phi(\mathbf{x}) \to s:\psi(\mathbf{x}^{\to g})$

The translation shows that there is a fundamental difference in the way mappings are interpreted in C-OWL and in calvanese2002a. While C-OWL mappings correspond to a universally quantified reading (Figure 1 a), calvanese2002a mappings have an existentially quantified readings (Figure 1 b/d). We will come back to this difference later.

17.2.3 DL for Information Integration (DLII)

A slightly different approach to the integration of different DL models is described in [Calvanese *et al.*, 2002a]. This approach assumes a partial overlap between the domains of the models M_i and M_j, rather than a complete embedding of them in a global domain. This is captured by the interpretation constraint OD_{ij}. The other assumptions (rigid designators and global inconsistency) are the same as for calvanese2002a.

An interpretation $\mathcal{I}$ associates to each M_i a domain Δ_i. These different models are connected by interschema assertions. Satisfiability of interschema assertions is defined as follows [3]

Definition 58 (Satisfiability of interschema assertions). *If $\mathcal{I}$ is an interpretation for M_i and M_j we say that $\mathcal{I}$ satisfies the interschema assertion*

[3]To simplify the definition we introduce the notation $\top^{\mathcal{I}}_{nij} = \top^{\mathcal{I}}_{ni} \cap \top^{\mathcal{I}}_{nj}$ for any $n \geq 1$. Notice that $\top^{\mathcal{I}}_{nij} = \Delta^n_i \cap \Delta^n_j$.

$\phi \sqsubseteq_{ext} \psi$, *if* $\phi^{\mathcal{I}} \subseteq \psi^{\mathcal{I}}$ $\phi \not\sqsubseteq_{ext} \psi$, *if* $\phi^{\mathcal{I}} \not\subseteq \psi^{\mathcal{I}}$
$\phi \equiv_{ext} \psi$, *if* $\phi^{\mathcal{I}} = \psi^{\mathcal{I}}$ $\phi \not\equiv_{ext} \psi$, *if* $\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}$
$\phi \sqsubseteq_{int} \psi$, *if* $\phi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij} \subseteq \psi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij}$
$\phi \equiv_{int} \psi$, *if* $\phi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij} = \psi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij}$
$\phi \not\sqsubseteq_{int} \psi$, *if* $\phi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij} \not\subseteq \psi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij}$
$\phi \not\equiv_{int} \psi$, *if* $\phi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij} \neq \psi^{\mathcal{I}} \cap \top^{\mathcal{I}}_{nij}$

As before $\equiv_{ext}$ and $\equiv_{int}$ are definable as conjunctions of $\sqsubseteq_{ext}$ and $\sqsubseteq_{int}$, so we can ignore them for the DFOL translation. Furthermore, a distinction is made between extensional and intentional interpretation of interschema assertions, which leads to different translations into DFOL.

interschema assertions	**DFOL**
$\phi \sqsubseteq_{ext} \psi$	$i : \phi(\mathbf{x}) \rightarrow j : \psi(\mathbf{x}^{i\rightarrow})$
$\phi \equiv_{ext} \psi$	$i : \phi(\mathbf{x}) \rightarrow j : \psi(\mathbf{x}^{i\rightarrow})$ and $j : \psi(\mathbf{x}) \rightarrow i : \phi(\mathbf{x}^{j\rightarrow})$
$\phi \not\sqsubseteq_{ext} \psi, \phi \not\equiv_{ext} \psi$	No translation
$\phi \sqsubseteq_{int} \psi$	$i : \phi(\mathbf{x}^{\rightarrow j}) \rightarrow j : \psi(\mathbf{x})$
$\phi \equiv_{int} \psi$	$i : \phi(\mathbf{x}^{\rightarrow j}) \rightarrow j : \psi(\mathbf{x})$ and $j : \psi(\mathbf{x}^{\rightarrow i}) \rightarrow i : \phi(\mathbf{x})$
$\phi \not\sqsubseteq_{int} \psi, \phi \not\equiv_{int} \psi$	No translation

While the extensional interpretation corresponds to the semantics of mappings in calvanese2002a, the intentional interpretation corresponds to the semantics of mappings in C-OWL. Thus using the distinction made in this approach we get an explanation of different conceptualizations underlying the semantics of C-OWL and calvanese2002a that use an extensional and an intentional interpretation, respectively.

17.2.4 ϵ-connections

A different approach for defining relations between DL knowledge bases has emerged from the investigation of so-called ϵ-connections between abstract description systems [Kutz *et al.*, 2004]. Originally intended to extend the decidability of DL models by partitioning them into a set of models that use a weaker logic, the approach has recently been proposed as a framework for defining mappings between ontologies [Grau *et al.*, 2004].

In the ϵ-connections framework, for every pair of ontologies ij there is a set ϵ_{ij} of *links*, which represents binary relations between the domain of the i-th ontology and the domain of the j-th ontology. Links from i to j can be used to define i-concepts, in a way that is analogous to how roles are used to define concepts. In the following table we report the syntax and the semantics of i-concept definitions based on links. (E denotes a link from i to j. The only assumption about the relation between domains is global inconsistency, see above).

In DFOL we have only one single relation from i to j, while in ϵ-connection there are many possible relations. However, we can use a similar trick as used in [Borgida and Serafini, 2003b] to map relations to interschema relations: each of the relations in ϵ_{ij} acts as a r_{ij}. To represent ϵ-connections it is therefore enough to label each arrow variable with the proper link name. The arrow variable $x^{\xrightarrow{E} i}$ is read as the arrow variable $x^{\rightarrow i}$ where r_{ij} is intended to be the interpretation of relation E_{ij}. With this syntactic extension of DFOL, concept definitions based on links (denoted as E) can be codified in DFOL as follows:

ϵ-connections	**DFOL**
$\phi \sqsubseteq \exists E.\psi$	$i:\phi(x) \rightarrow j:\psi(x^{i\xrightarrow{E}})$
$\phi \sqsubseteq \forall E.\psi$	$i:\phi(x^{\xrightarrow{E}j}) \rightarrow j:\psi(x)$
$\phi \sqsubseteq \geq nE.\psi$	$i:\bigwedge_{k=1}^{n} \phi(x_1) \rightarrow$ $j:\bigwedge_{k\neq h=1}^{n} \psi(x_k^{i\xrightarrow{E}}) \wedge x_k \neq x_h$
$\phi \sqsubseteq \leq nE.\psi$	$i:\phi(x) \wedge \bigwedge_{k=1}^{n+1} x = x_k^{\xrightarrow{E}j} \rightarrow$ $j:\bigvee_{k=1}^{n+1} \left(\psi(x_k) \supset \bigvee_{h\neq k} x_h = x_k\right)$

We see that like calvanese2002a, links in the ϵ-connections framework have an extensional interpretation. The fact, that the framework distinguishes between different types of domain relations, however, makes it different from all other approaches.

Another difference to the previous approaches is that new links can be defined on the basis of existing links, similar to complex roles in DL. Syntax and semantics for link constructors is defined in the usual way: $\left(E^{-1}\right)^{\mathcal{I}} = \left(E^{\mathcal{I}}\right)^{-1}$ (Inverse), $(E \sqcap F)^{\mathcal{I}} = E^{\mathcal{I}} \cap F^{\mathcal{I}}$ (Conjunction), $(E \sqcup F)^{\mathcal{I}} = E^{\mathcal{I}} \cup F^{\mathcal{I}}$ (Disjunction), and $(\neg E)^{\mathcal{I}} = (\Delta_i \times \Delta_j) \setminus E^{\mathcal{I}}$ (Complement). Notice that, by means of inverse links we can define mappings of the b and d type. E.g., the ϵ-connection statement $\phi \sqsubseteq \exists E^{-1}\psi$ corresponds to the DFOL bridge rule $i:\phi(x) \rightarrow j:\psi(x^{i\rightarrow})$ which is of type b). Similarly the ϵ-connection $\phi \sqsubseteq \forall E^{-1}\psi$ corresponds to a mapping of type d).

As the distinctions between different types of links is only made on the model theoretic level, it is not possible to model Boolean combinations of links. Inverse links, however, can be represented by the following axiom:

$$i:y = x^{\xrightarrow{E}j} \rightarrow j:y^{\xrightarrow{E^{-1}}i} = x$$
$$j:y^{\xrightarrow{E^{-1}}i} = x \rightarrow i:y = x^{\xrightarrow{E}j}$$

Finally the inclusion axioms between links, i.e., axioms of the form $E \sqsubseteq F$ where E and F are homogeneous links, i.e., links of the same ϵ_{ij}, can be translated in DFOL as follows:

$$i:x = y^{\xrightarrow{E}j} \rightarrow j:x^{i\xrightarrow{F}} = y$$

We can say that the ϵ-connections framework significantly differs from the other approaches in terms of the possibilities to define and combine mappings of different types.

17.3 Discussion and Chapter Conclusions

The encoding of different mapping approaches in a common framework has two immediate advantages. The first one is the ability to reason across the different frameworks. This can be done on the basis of the DFOL translation of the different approaches using the sound and complete calculus for DFOL [Ghidini and Serafini, 1998]. As there are not always complete translations, this approach does not cover all aspects of the different approaches, but as shown above, we can capture most aspects. There are only two aspects which cannot be represented in DFOL, namely "non mappings" ($i:\phi \xrightarrow{*} j:\psi$ in C-OWL, $\phi \not\sqsubseteq_{int} \psi$ etc. in DLII) and "complex mappings" such as complex links in ϵ-connection. The second benefit is the possibility to compare the expressiveness of the approaches. We have several dimensions along which the framework can differ:

Arity of mapped items[4] C-OWL allows only to align constants, concepts and roles (2-arity relations), ϵ-connections allow to align only 1-arity items, i.e., concepts, while DLII and calvanese2002a allow to integrate n-arity items.

Positive/negative mappings Most approaches state positive facts about mapping, e.g that two elements are equivalent. The DLII and C-OWL frameworks also allow to state that two elements do not map ($\phi \not\equiv \psi$).

Domain relations The approaches make different assumptions about the nature of the domain. While C-OWL and ϵ-connections do not assume any relation between the domains, DLII assumes overlapping domains and calvanese2002a assumes local domains that are embedded in a global domain.

Multiple mappings Approaches with multiple mappings allow different kind of mapping relations. Only the ϵ-connection approach supports the definition of different types of mappings between ontologies. In general multiple mappings need for more than one domain relation.

Local inconsistency Some approaches provide a consistent semantics also in the case in which some of the ontologies or mappings are inconsistent.

We summarize the comparison in the following table.

	Int. constr. (cf. fig. 17.1)				Mapping type			Domain relation	Arity	Local $\perp$
	a)	b)	c)	d)	Pos.	Neg.	Mult.			
C-OWL	$\times$			$\times$	$\times$	$\times$		Het.	2	$\times$
calvanese2002a		$\times$		$\times$	$\times$			Incl.	n	
DLII	$\times$	$\times$			$\times$	$\times$		Emb.	n	
ϵ-Conn.	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	Het.	1	

We conclude that existing approaches make choices along a number of dimensions. These choices are obviously influenced by the intended use. Approaches intended for database integration for example will support the mapping of n-ary items that correspond to tuples in the relational model. Despite this fact, almost no work has been done on charting the landscape of choices to be made when designing a mapping approach, and for adapting the approach to the requirements of an application. The work reported in this chapter provides the basis for this kind of work by identifying the possible choices on a formal level. An important topic of future work is to identify possible combinations of features for mapping languages on a formal level in order to get a more complete picture of the design space of mapping languages.

Chapter 18

Categorical presentation of alignment and merge

In this section we propose a special case of what we said above and an application to the problem of generating an ontology o_m which mediates between two heterogeneous ontologies o and o'.

Ontology merging describes the process of integrating two (or more) ontologies into a single one. How this is done best is a subject of ongoing research in the Semantic Web community. In this chapter, we propose a generic solution to the question, what the result of a merging should be in the ideal case. We do this independent of a specific choice of ontology representation language, and thus provide a sort of *blueprint* for the development of algorithms applicable in practice. Our methods are taken from category theory. More precisely, we argue that ontology merging is best captured by the notion of categorical *pushout*. This presentation is a first step towards the development of practically applicable algorithms.

In this chapter we explain how merging of ontologies is captured by the pushout construction from category theory, and argue that this is a very natural approach to the problem. For this purpose, we view category theory as a universal "meta specification language" that enables us to specify properties of ontological relationships and constructions in a way that does not depend on any particular implementation. This can be achieved since the basic objects of study in category theory are the *relationships* between multiple ontological specifications, not the internal structure of a single knowledge representation.

Categorical pushouts are already considered in some approaches to ontology research [Jannink *et al.*, 1998; Kent, 2000; Schorlemmer *et al.*, 2002] and we do not claim our treatment to be entirely original. Still we have the impression that the potential of category theoretic approaches is by far not exhausted in todays ontology research. Consequently, our goal is not only to demonstrate how a concrete problem can be captured with categorical formalisms, but also to give introduction and motivation for those who did not study the mathematical framework of category theory yet. In this respect our attempts to make categories more accessible follow the spirit of [Goguen, 1991] and the current discussions on the *Information Flow Framework* [Kent, 2000].

In contrast to some of the works mentioned above, we do not try to give a comprehensive overview of even the most important categorical methods. Instead, our treatment will focus on the particular aspect of ontology merging, for which we will give both intuitive explanations and precise definitions. This reflects our belief that, at the current stage of research, it is not desirable to fade out the mathematical details of the categorical approach completely, since the interfaces to current techniques in ontology research are not yet available to their full extent. We will also keep this treatment rather general, not narrowing the discussion to specific formalisms — this added generality is one of the strengths of category theory.

We proceed as follows: In the next section, we will give the intuitions that make a categorical framework fitting the problems of ontology alignment and merging. Then we provide a short introduction to categories, together with some basic examples that we will consider throughout this text. Then we investigate how category theory deals with cartesian products and relations, which we will utilize to model "ontology mapping" and "ontology alignment" in categorical terms thus establishing the framework for the first part of any merging operation. Section 18.3 then forms the core of this note, explaining pushouts and their relevance to ontology merging. In Section 18.7 we will explain how our theoretical considerations can be used to obtain practical methods for ontology merging. The last section includes references to the literature, pointing to sources of further information on categorical and ontological issues touched on herein.

18.1 Intuition

We present the intuition behind alignment and merging in function of some approximation relation between ontologies. We give the classical interpretation of this in both model theoretic terms and categorical terms. This is informally presented in Figure 18.1.

Let us call approximation a relation between ontologies which expresses that one ontology (a) is a representation of at least the same modeled domains as another ($\alpha(o, o')$). In logic, this relation corresponds to entailment. In category theory, the ontology will be called an *object* and the relation a *morphism*. This formulation will be completed below, but one can define other relations between ontologies such as having at least one common approximated ontology. Syntactically, it is possible to provide a set of generators that will complete an ontology (e.g., adding a constraint on a class, classifying an individual), providing an approximated ontology.

Model-theoretic semantics assigns to any ontology the set of its models. If the ontology is correctly designed, the modeled domain is part of these. Model-theoretic semantics provides a formal meaning to the intuitions behind notions such as approximation: an ontology approximates another if its models contains all the models of the other (this is the standard interpretation of entailment). So, the more approximated an ontology, the less models it has[1].

In these very general terms, aligning two ontologies (o and o') consists of finding a most specific ontology ($\alpha(o, o')$) that approximates both ontologies. If one ontology is approximated by another, the result of alignment should be the latter ($\alpha(o, o') = o$). In model-theoretic terms, it amounts to finding an ontology whose set of models is maximal for inclusion and is included in the intersection of the set of models of the two aligned ontologies. In categorical terms, it means that there exists an object ($\alpha(o, o')$) and a pair of morphisms from it to the two ontologies (o and o').

Finding the alignments between two ontologies is very useful. In particular, if one wants to merge two ontologies ($\mu(o, o')$), it is sufficient to stick the non aligned part of one ontology to the aligned subpart with the other ontology. We will see in the remainder that this corresponds, in categorical terms to the push-out construction.

[1]This amounts to consider the image in the domain of interpretation as a model, not the interpretation function itself, of course.

$$\begin{tikzcd}
 & & & & & \cdots \\
 & & & & m \arrow[ur] & \\
 & & o' \arrow[r] & \mu(o,o') \arrow[ur] & & \\
 & \alpha(o,o') \arrow[ur] \arrow[r] & o \arrow[ur] & & & \\
a \arrow[ur] & & & & &
\end{tikzcd}$$

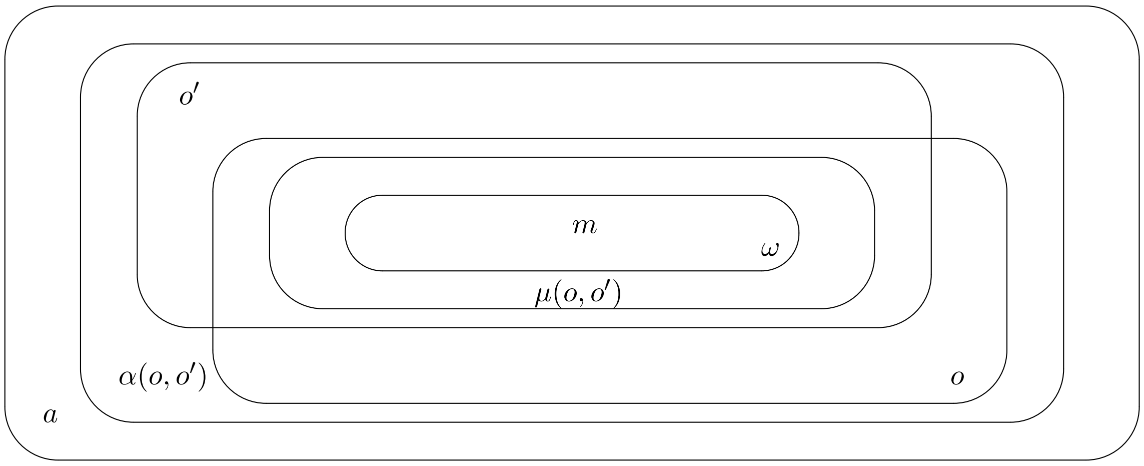


Figure 18.1: Relations between ontologies and alignment ($\alpha(o, o')$) and the corresponding model-theoretic interpretation (each ontology is represented by its set of models).

This general description of alignment, can be compared to the three cases introduced. Indeed, if taken literally

coverage corresponds to the point where $\alpha(o, o') = \emptyset$ ($\emptyset$ being the empty ontology);
granularity corresponds to the case $\alpha(o, o') = o$;
perspective corresponds to the general case presented here.

Of course, the reality is that the most frequent cases do not distinctly belong to one of these.

In its most general form, the term "ontology alignment" can refer to almost any formal description of the (semantic) relationship between ontologies. Deliverable D2.2.1 [Bouquet *et al.*, 2004a] discussed a more restricted conception of the term, that conceived alignments as pairs of elements of the ontologies[2], together with meta-information on the type of the relation and the confidence in its correctness.

An ontology alignment is thus given a set theoretic definition as a set of mappings. Though perfectly acceptable in many applications, it has the disadvantage of using entities—something local—as the basis of ontology alignment—something global. Indeed, the complexity of ontology languages like OWL (see [Antoniou and van Harmelen, 2004] for an introduction) enables us to express more complicated relationships between elements even without the use of meta-logical constructions. For example, consider an ontology with a concept $Professor$ and one with the concepts $Woman$ and Man. A comprehensive alignment should contain the information that $Professor$ is a subconcept of the union of $Woman$ and Man. On the one hand, this statement has a non-local flavor in the sense that it cannot be expressed in terms of relations between pairs of (atomic) elements. On the other hand, we deal with a relationship between concept expressions which is not a mere equality of two statements. In contrast, most of todays alignment algorithms focus on equality or similarity mappings between pairs of elements.

Here, however, we want to discuss an approach that emphasizes alignment information which can be represented in terms of the ontology language. This has the advantage that alignments are more closely related to the ontological formalism under consideration, thereby allowing for simple and concrete descriptions of the merging of aligned ontologies. On the other hand, while the relationships between ontologies can be structurally more complex, meta-level information like confidence values are not included in the framework[3]. Our presentation will be simplified by using a coherent notation that employs elementary *category theory* as a concise meta-language.

After presenting some related work, we start our investigation of so-called *V-alignments* in Section 18.4. We give a formal definition, introduce merging, and present an algebra for working with these simple alignments. Thereafter, in Section 18.5, we extend our approach to a more complex type of alignments called *W-alignments*. Again we present suitable formalizations of merging and composition within this framework. Section 18.6 shows how alignment in the disjunctive first-order logic from Chapter 17 can be expressed by means of W-alignments. Finally, Section **??** gives some concluding remarks.

Related work

In [Verheijen *et al.*, 1998], the categorical approach is mentioned but not well formalized. [Bench-Capon and Malcolm, 1999] use morphisms of algebraic specifications (see [Guttag and Horning,

[2]We speak of "elements of an ontology" to refer to arbitrary semantic entities of a given ontology language, like e.g. concepts, relations, or instances. Comparisons of elements in different ontologies are restricted to elements of similar type, i.e. we would never compare instances with relations etc.

[3]Though they could be added "on top" if desired.

1978]) to define morphisms between ontologies and say a *relation* (an alignment in their sense) between ontologies O_1 and O_2 consists of an ontology O and a pair of morphisms $\chi_1 : O \rightarrow O_1$ and $\chi_2 : O \rightarrow O_2$. This is precisely the definition of alignments we use, but they do not provide any means of representing complex alignments as we do. In [Kent, 2000], a category theoretic approach using the information flow theory of Barwise and Seligman [Barwise and Seligman, 1997] is given, with no concrete representation of alignments. Information Flow is also the basis of an implemented system called IF-Map [Kalfoglou and Schorlemmer, 2002] designed for automated ontology mapping, but they do not have a categorical representation for rich alignments. [Hitzler *et al.*, 2005b] gives a concrete example of a representation of an alignment in category theory, but since it is so simplistic, it is hard to see the generality of the approach. Much more details on the categorical approach are given in a survey on ontology mapping [Kalfoglou and Schorlemmer, 2005], in particular in Sections 2.a. and 3.f.

18.2 Categorical preliminaries

We will now express this intuition more formally in the language of category theory. For that purpose, we introduce briefly its preliminaries. This section be skipped by the knowledgeable reader.

Given two ontologies, it is for example possible that one is a sub-ontology of the other or that both of them represent the same information, i.e. that they are equivalent. Depending on the chosen representation of ontologies this can be recognized by looking at the internal structures of the given ontologies. But looking at the internal structures requires an individual treatment for each new approach to the mathematical modelling of ontologies, while a generic treatment which abstracts from the particular choice of ontology language would certainly be preferable for understanding what the result of a merging shall be.

Fortunately, there is another possibility to compare objects mathematically, which lends itself to such a generic treatment. For example, two partially ordered sets can be considered to be *equivalent*, if there exists a bijective function (i.e. one which is one-to-one and onto) between these sets which does also preserve the order (i.e. which is *monotonic*). In this case, being monotonic means that a function respects the internal structure of partially ordered sets, while bijectivity indicates the equivalence of two ordered sets. Structure-preserving functions are a typical implementation of what is called a *morphism* in category theory, and what we will recognize as a suitable substitute for the consideration of internal structures.

While monotonic functions are reasonable morphisms for comparing partial orders, other mathematical spaces may suggest different kinds of morphisms: Vector spaces are considered with linear functions, groups with group homomorphisms, geometries with movements (e.g. on the plane), topological spaces with continuous functions, etc. Considering plain sets (with no further internal structure), a morphism between two sets could be any function between them. This approach ignores most individual features of the elements of a set: functions do not distinguish whether the elements of some set are labeled a, b, c, or dog, cat, $house$. Labels are only needed to specify the function, but the essential feature of a set turns out to be its cardinality. Sets of the same size would therefore appear equivalent.

The idea that emerges from these observations is that the relationships between objects are basically captured by the morphisms that exist between them. By deciding for a particular type of morphisms, we determine which internal properties of the mathematical objects are considered "essential" (e.g. order structure or cardinality). This is the approach taken in category theory: a class of *objects* (e.g. order structures) is equipped with *morphisms* (e.g. monotonic functions), thus forming a large directed graph with objects as nodes and morphisms as arrows. Depending on the given situation, arrows can be identified with certain functions or relations between the entities that were chosen for objects, but no such concrete meaning is required. In order to constitute a category, a directed graph only has to include a *composition* operation for pairs of compatible arrows, satisfying some straightforward axioms that are typical for the composition of functions and the relational product. Let us now make this informal description precise.

Definition 59 (Category)**.** *A category $\mathscr{C}$ consists of the following:*

– *A class*[4] *of* objects $|\mathscr{C}|$*,*
– *for any two objects A, $B \in |\mathscr{C}|$, a set $\mathscr{C}(A, B)$ of* morphisms *from A to B,*
– *for any three objects A, B, $C \in |\mathscr{C}|$, a* composition function

$$\circ : \mathscr{C}(B, C) \times \mathscr{C}(A, B) \to \mathscr{C}(A, C),$$

that combines a morphism from A to B with one from B to C to obtain a morphism from A to C,
– *for any object $A \in |\mathscr{C}|$, an* identity morphism $\mathrm{id}_A \in \mathscr{C}(A, A)$.

This data is required to satisfy the following additional axioms:

– *For all $f \in \mathscr{C}(A, B)$, $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$, and*
– *for all $f \in \mathscr{C}(A, B)$, $g \in \mathscr{C}(B, C)$, and $h \in \mathscr{C}(C, D)$, $((h \circ g) \circ f) = (h \circ (g \circ f))$.*

We will also write $f : A \to B$ for $f \in \mathscr{C}(A, B)$.

The additional requirements for a directed graph to become a category are few indeed, and in many cases the definition of morphisms already entails an obvious and well-behaved composition operation. Yet the results one can derive from the components of a category are surprisingly rich, and usually all the essential knowledge about a class of objects is captured by some suitable category. The defining axioms of a category provide an abstract interface to all kinds of structures, and category theory allows for a unified treatment of all of them, since internal features of objects are disregarded completely.

As a simple example, consider the category Set of all sets and functions between them, i.e. $|\mathsf{Set}|$ consists of all sets, and given two sets $A, B \in |\mathsf{Set}|$ the collection $\mathsf{Set}(A, B)$ of all morphisms from A to B is just the collection of all functions from A to B. The identity morphisms are given as the identity functions, i.e. those functions which map all elements onto themselves. Composition in Set is given by composition of functions. Another category which we will discuss in more detail later is the category Poset of all partially ordered sets together with monotone functions. We remark that categorical morphisms are often given by functions, but that this is by no means necessary. For example, we can view a single partially ordered set as a category, where we have a single morphism between two elements p and q if and only if $p \leq q$. Composition is provided by transitivity and identity morphisms exist by reflexivity. This last example might appear somewhat peculiar at first, but it shows how general the basic notions in category theory really are.

Above, we gave two examples of specific relationships between objects: being a subobject and being equivalent. In the given examples, these do still refer to the internal structure of the objects,

[4]*Class* should be understood as a kind of *collection*. Classes of objects are not always *sets* of objects for reasons which have to do with Russell's paradox from set theory, but we shall not inconvenience us with such matters here. The term *class* is certainly not supposed to mean classes as e.g. in Description Logics!

so we have to consider defining both in purely categorical terms. Let us first explore the notion of equivalence (or, speaking categorically, *isomorphism*) for the categories Set and Poset. Restating our earlier insights, we find that two partially ordered sets P and Q are equivalent (isomorphic) whenever there is a monotone function $f : P \to Q$ that has a monotone *inverse*, i.e. for which there is a monotone function $g : Q \to P$ with $g \circ f = \mathrm{id}_P$ and $f \circ g = \mathrm{id}_Q$. Generalizing this to arbitrary categories, we call a morphism an *isomorphism* if it has a (necessarily unique) inverse morphism.

Application of this definition to Set reveals bijective functions as the isomorphisms of sets. Likewise, the categorical definition immediately provides us with a suitable notion of equivalence in any category we may wish to study. As mentioned in the introduction, the possible translations of information between ontologies are suggestive morphisms for ontology research. Indeed, no matter how ontologies and translations between them are implemented, composition of translations methods and the existence of identity translations should always be available. In consequence, isomorphic ontologies are intuitively described by the possibility of translating knowledge back and forth between them without loosing information. In a similar fashion, we will gain a general description of *ontology merging* later on, though it will be a bit more involved.

To obtain the notion of a subobject, let us consider the category Set and note that every subset $A \subseteq B$ of a given set B can be obtained as the image (range) of some injective (i.e. one-to-one) function into B. By injectivity, the domain of any such function is in bijective correspondence with the subset A. Thus any subset can be given by some injective function and any such function defines a subset[5]. We remark that the subobjects are really given by the injective function, not by its domain: for instance, a function from the one-element set $\{a\}$ to the natural numbers can map a to 0, to 1, or to any other number. In spite of the constant function domain, this always denotes different subsets ($\{0\}$, $\{1\}, \dots$) of the natural numbers. Yet, due to injectivity, the set $\{a\}$ is surely isomorphic (thus essentially equivalent) to the indicated subsets; still the *position* of the isomorphic copy of $\{a\}$ as a subobjects of the codomain does make a difference.

Now in order to obtain a categorical definition of injectivity, we observe that an injective function $f : A \to B$ in Set has the following peculiar property: for every pair of functions $g : C \to A$ and $h : C \to A$, the equality $f \circ g = f \circ h$ implies $g = h$. Morphisms (of arbitrary categories) with this feature are called *monomorphisms* and it turns out to be appropriate to consider a monomorphism as the specification of a subobject of its codomain. Intuitively, the condition describes a monomorphism $f : A \to B$ as an embedding of A into B that does not obliterate any essential features of A. Two morphisms $C \to A$ that are distinct on some part of A must also be distinct when extended via f to B. Thus the monomorphism can be thought of as a pointer to the specified subobject, which in turn is isomorphic to the domain of the monomorphism.

These examples give but a brief glimpse at the expressiveness of category theory. We continue next with discussing some constructions which will help in understanding pushouts.

Products and Relations

In set theory, the cartesian product of two sets is defined as the set of all pairs of elements from two given sets. This is not a suitable description from the viewpoint of category theory, since we want to avoid to mention the internal (element-based) structure of our objects. In order to rephrase this in categorical language, we need to find alternative criteria that rely exclusively on properties of

[5]Note, however, that there are usually many injective functions with the same image. So for obtaining an exact definition of subobjects, one would still have to identify the equivalent injective functions.

the morphisms. To this end, an important observation is that a product does in general also provide two *projection functions* to the first respectively second component of the product. Furthermore, the product is distinguished by a *universal property* given in the next definition.

Definition 60 (Product)**.** *Consider a category $\mathscr{C}$ and objects A, $B \in |\mathscr{C}|$. Given an object $C \in |\mathscr{C}|$ and morphisms $p_1 : C \to A$ and $p_2 : C \to B$, we say that (C, p_1, p_2) is the* product *of A and B if the following universal property holds:*

For any object $D \in |\mathscr{C}|$ and morphisms $q_1 : C \to A$ and $q_2 : C \to B$, there is a unique morphism $\langle q_1, q_2\rangle : D \to C$, such that $q_1 = p_1 \circ \langle q_1, q_2\rangle$ and $q_2 = p_2 \circ \langle q_1, q_2\rangle$. The latter situation is depicted in the following diagram:

CURRENTLY UNAVAILABLE

For example, when considering the category Set and its usual cartesian product, we can define the function $\langle q_1, q_2\rangle$ by setting $\langle q_1, q_2\rangle(d) = (q_1(d), q_2(d))$. In spite of this, the above defines the cartesian product of sets only up to isomorphism (i.e. bijective correspondence) — *any* set with the cardinality of the cartesian product can be equipped with appropriate morphisms. This is a typical feature of category theory: isomorphic objects are not distinguished, since they behave similar in all practical situations. It is the choice of morphisms that determines what distinctions are considered relevant in the first place. Yet we will henceforth assume that we have fixed one representative for the product of any two elements A and B which we label $A \times B$. We also remark that products do not exist in every category, so the previous convention needs to be restricted to existing products.

We remark, nevertheless, that the notion of *product* of two objects depends solely on the chosen category, i.e. on the objects and their morphisms. Fixing, for example, a specific ontology language, and finding an agreement on which features of an ontology should be preserved by a corresponding morphism, we obtain a notion of *product* in a canonical way.

The categorical product definition also turns out to be suitable to model many well-known product constructions. As a product of two partially ordered sets one usually considers the product-order, i.e. the cartesian product of the two sets, ordered such that a pair (a, b) is below a pair (c, d) whenever a is below c and b is below d. The partially ordered set obtained in this way corresponds to the categorical product in Poset, which arguably is the reason for the significance of this particular construction. To give another example: If we consider a single partially ordered set as a category, as discussed earlier, then the product of two of its elements is just the greatest lower bound. This is also an example where a product may fail to exist.

Combining the product construction with our earlier considerations on subobjects into practice, we can also introduce *binary relations* on objects. Indeed an ordinary set-theoretic binary relation is just a subset of the cartesian product of two objects. Hence it makes sense to consider a monomorphism $r : D \to (A \times B)$ from some object D to the product of A and B as a binary relation between A and B. Note that this does also give us two functions $p_1 \circ r : D \to A$ and $p_2 \circ r : D \to B$ to the two components of the product, for which the morphism r is already the unique factorization that exists due to the definition of a product. Much generalized theory can be developed around this, but we shall be content at this point.

18.3 Merging ontologies via pushouts

We will now return to our initial motivation. Our intuition is that the objects of our category represent ontologies and that the morphisms between them serve as meaningful transitions between these specifications. The categorical product construction is not suitable for the purpose of modelling ontology merging, since it does obviously not consider any relationship between two ontologies. Such a relationship — commonly referred to as an *ontology mapping* — however is the base of an ontology merging process, so we have to find a means of modelling it in our categorical setting. We are in fact more interested in a certain kind of *sum* than in a product. Indeed, if two ontologies were entirely unrelated, they could be combined by just taking their disjoint union (provided that this operation makes sense for the chosen ontology representation language). However, we are more interested in merging ontologies that do overlap (via some mapping), where some elements are related while others are not. Merging two such ontologies should lead to a new ontology that identifies equivalent elements but that tries to keep unrelated elements apart, as far as this is possible without violating the requirements that are imposed on the structure of an ontology.

As an example, let us consider the following two partial orders:

CURRENTLY UNAVAILABLE

We assume that some elements of these structures are known to be equivalent. This is expressed by a relation $R \subseteq P \times Q$ (called an ontology alignment above) that we define as the set of pairs:

$$R = \{\langle a, 1\rangle, \langle b, 2\rangle, \langle c, 4\rangle, \langle f, 5\rangle, \langle g, 3\rangle\}.$$

This relation is not an object of the category, but it can easily be expressed as the poset $\alpha(P, Q)$:

CURRENTLY UNAVAILABLE

and a pair of morphisms from $\alpha(P, Q)$ to P and Q mapping each element from $\alpha(P, Q)$ to the corresponding element in P and Q (here, the labels which have been ascribed to the elements are not meaningful, it is possible to erase them, the morphisms play the central role). It is clear that these morphisms preserve the order.

A reasonable result of merging the posets P and Q would then be the following structure:

CURRENTLY UNAVAILABLE

Observe that all elements related by R are indeed identified, but that some additional identifications are necessary to obtain a partially ordered set. Categorically, we can already specify the data that we have considered for such an operation. The given situation is depicted in the following diagram:

CURRENTLY UNAVAILABLE

The shape of the arrow from R to $P \times Q$ indicates that it defines a subobject (a monomorphism). The dotted arrows r_1 and r_2 are those that are obtained by composing the projections of the product with this monomorphism. They project every pair of elements of R to its first and second component, respectively. Now the result of merging P and Q is not just some poset $\mathsf{merge}_R(P, Q)$, but also the two obvious embeddings of P and Q into $\mathsf{merge}_R(P, Q)$. As a diagram, we obtain:

CURRENTLY UNAVAILABLE

The property that R-related elements are identified can now be expressed in terms of functions: we find that, for any pair $(p, q) \in R$, $e_1(p) = e_2(q)$. Still a better way to express this for arbitrary morphisms is to say that $e_1 \circ r_1 = e_2 \circ r_2$.

This condition alone, however, does not suffice. Usually, there are many objects for which $e_1 \circ r_1 = e_2 \circ r_2$ holds. Which of these is the one which we want to consider as the *merging* of P and Q? Clearly, the merging shall not identify anything unnecessarily. This can be stated by means of another *universal property*, as follows.

Definition 61 (Pushout)**.** *For a category $\mathscr{C}$, consider objects R, P, Q, and morphisms $p_1 : R \to P$ and $p_2 : R \to Q$. An object S together with two morphisms $e_1 : P \to S$ and $e_2 : Q \to S$ is a* pushout *if it satisfies the following properties:*

(i) $e_1 \circ p_1 = e_2 \circ p_2$, *i.e. the following diagram* commutes

CURRENTLY UNAVAILABLE

Condition (ii) in this definition states the universal property of the pushout, requiring that it is in a sense the most general object that meets all requirements. Let us try to explain this a bit further. We have already understood that in this setting we can encode the ontology mapping (e.g. binary relation) R conveniently, in that the resulting S identifies (at least) all those elements which are related by R. But now we want to *avoid* the identification of other elements as much as possible. Intuitively, this means that a suitable pushout object needs to keep elements from both components as distinct as possible, while still implementing all necessary identifications, and without including irrelevant information. Enforcing the desired identifications was achieved by condition (i) in the above definition. Excessive identifications are prevented by requiring the *existence* of a factorization m: appending m to e_1 and e_2 cannot make prior identifications undone, and hence a pair that was merged in S can never be separated in an alternative solution (T, f_1, f_2) if a suitable m is known to exist. Finally, the possibility of including entirely unrelated information, like adding some elements not present in either P or Q, is ruled out by assuring *uniqueness* of the factorization m: if S would include elements that are neither in the image of e_1 nor in the image of e_2 then a valid factorization can assign these to arbitrary values in T without loosing the factorization property — but this would result in many possible choices in place of m. In other words, having "unnecessary" elements in the S would result in additional degrees of freedom in the choice of m, thus violating the required uniqueness.

Note also that we ignored our earlier restriction of R being a subobject of the product $P \times Q$. However, by the universal property of the product, any object R with functions to P and Q must have a unique factorization through the product, and hence does still capture part of the idea of a relation. Furthermore, such a generalized R can also be viewed as a suitable background knowledge that both P and Q are based on. In spite of this generalization, R is still an object of the considered category, i.e. it is itself an ontology with all necessary structure. We just dropped some side conditions on this object, such that some redundancy can be introduced into the ontology mapping if desired.

18.4 Simple Alignments

In spite of our declared objective to express alignments in a non-local fashion, we will start our considerations with a very simple relational type of alignments. Probably the most basic way to describe relationships between two ontologies is to identify those elements which represent the same semantic entities. This can be adequately described by a binary relation between the sets of elements.

To simplify our presentation, we employ a unified notation based on elementary category theory. Category theory describes objects (in our case: ontologies) and their relationships in a leveled fashion:

– On the concrete level, one starts with a given collection of objects and defines directed relationships (called *morphisms*) between them. The latter requires to refer to the details of the objects structure. For example, in Section 18.4.1, we introduce *ontology refinements*—a special type of functions—as morphisms between ontologies.

– On the abstract level, one works solely on the structure of objects and morphisms that are defined in each concrete case. Properties and constructions on the abstract level can thereby be carried out without detailed knowledge about the concrete level, and yet one can produce meaningful results. In our given case, this enables us to define alignments without restricting to any concrete ontology language.

Thus category theory introduces a kind of "object-oriented" paradigm, that enable us to work on interfaces (to the unspecified concrete level) without knowing the implementation details (i.e. the ontology language that is in use). To see how morphisms can be used to externalize structural properties of ontologies, consider the sketchy description of simple alignments given above. The binary relation that describes a simple alignment is a set of pairs of elements, and we can assign to any such pair one element in the first and one element of the second ontology. These operations are called *projections* and can be visualized as follows:

CURRENTLY UNAVAILABLE

Here π_1 and π_2 are the according projection functions. Now we observe that any set A with two such projections can be interpreted as a relation, since the projections assign to any element of A a pair of elements. Thus the essence of being a binary relation is captured in the above diagram, even if we do not have any details about the sets A, O_1, and O_2. This view represents the abstract level of category theory: the shape of the diagram carries meaning, even if no further details about the concrete definitions are known.

18.4.1 Category theoretic alignments

In the rest of this work, some familiarity with category theory will be useful. However, we will usually provide examples that refer to problems on a concrete level, and the reader might be content with taking the categorical language as a somewhat more general formal description of the exemplified ideas. For more details on the basics of category theory, see [Pierce, 1991] for an easy yet good introduction. [Adámek *et al.*, 2004] gives something more elaborated.

Categories of ontologies

Most of our later considerations can nicely be done on the abstract level of category theory, but it is sensible first to introduce some underlying concrete levels that are meaningful in the context of ontology research. The objects that we will deal with here naturally are ontological descriptions, like description logic knowledge bases, first-order theories, or simple hierarchical taxonomies. However, we shall define morphisms only for ontology languages with a logical semantics that assigns to any knowledge base a collection of models (i.e. formal interpretations that respect the axioms and constraints of the knowledge base)[6].

Thus we can view ontologies as logical theories and define morphisms between them based on well-known studies in these fields. In particular, *theory morphisms* were discussed in *institution theory* [Goguen and Burstall, 1992] as suitable "translation functions" between logical theories. Basically, a theory morphism is a function from one ontology (regarded as a set of axioms) into another ontology, such that all of the axioms of the first ontology are mapped to statements that are satisfied in the second ontology. However, we only want to consider functions that respect the logical structure of the ontology language. This somewhat complicates the formal definition, since the ontologies that we compare might be based on different signatures (i.e. on different underlying alphabets). A formal definition employs the notion of *signature morphisms* known in institution theory. It can for example be found in [Goguen and Burstall, 1992, Definition 5].

For this work, it suffices to know that the intuition behind theory morphism is that they describe translation functions that "embed" logical theories into more specific theories. Thus any theory morphism describes a way to view some ontology as a generalization of another ontology, which is why the name "ontology refinement" is also appropriate for this type of morphism. In the following, we will work on the abstract level, with ontologies and their refinements as the concrete category that we have in mind.

V-alignments of ontologies

Assuming that a suitable concrete category of ontologies is available, we can now define simple ontology alignments on the abstract level. Due to the shape of the underlying diagram, we dub these alignments "V-alignments" in order not to confuse them with the informal idea of alignments in general.

Definition 62 (Ontology alignment)**.** *A* V-alignment *between two ontologies O_1 and O_2 is a triple $\langle O, p_1, p_2\rangle$ such that:*

- *O is an ontology (i.e.* an object *in the concrete category)*
- *$p_1 : O \to O_1$ and $p_2 : O \to O_2$ are ontology refinements (i.e. morphisms in the concrete category)*

For any V-alignment A, the object O in the definition is written $|A|$.

This very basic definition has already been used by several of the aforementioned works, e.g. [Bench-Capon and Malcolm, 1999; Kent, 2000; Kalfoglou and Schorlemmer, 2002; Hitzler *et*

[6]This is not very precise yet. However, all of the mentioned paradigms (which can be viewed as fragments of first-order logic), extensions with equality, and some typical higher order paradigms fall into the scope of this discussion. Nonmonotonic paradigms might require a different definition.

CURRENTLY UNAVAILABLE

Figure 18.2: Two ontologies modelling the same domain

al., 2005b], except that the terms "V-alignment" and "ontology refinement" are non-standard, and sometimes, the two morphisms are not explicitly described as theory morphisms in an institution.

Example 2. *In order to illustrate the definition, consider the ontologies in Figure 18.2, which will also serve us as a running example later on. The two ontologies describe simple taxonomies of concepts, but one is written in English while the other is in French.*

The ontologies given in Figure 18.2 are clearly related. For instance, one would like to express the fact that $Objet$ is equivalent to class $Object$, as well as $Vehicule$ is equivalent to $Vehicle$. In order to do so, let us define an alignment ontology O with concept names $\{Objet\text{-}Object$, $Vehicule\text{-}Vehicle$, $Hydravion\text{-}Seaplane$, $Maison\text{-}House$, $Residence\text{-}Domicile\}$ and no further axioms.[7]

To obtain the intended V-alignment, let $f_1 : O \to O_1$ and $f_2 : O \to O_2$ be the obvious projection mappings that map each concept of O to the concept given in the first respectively second part of its concept name. Taxonomies form a simple description logic, so they are expressed as theories in a concrete institution. In such institution, theory morphisms are exactly order-preserving functions. Thus, it is routine to check that f_1 and f_2 are indeed theory morphisms. For further reference, we label this V-alignment $A = \langle O, f_1, f_2 \rangle$. Note that the names of the concepts in O are arbitrary and were just chosen to simplify the presentation. In particular, they do not affect the content of the alignment.

Merging with simple alignments

Once a V-alignment between two ontologies is known, it is desirable to integrate the aligned ontologies into a combined knowledge base. This operation, called *ontology merging*, aims at uniting heterogeneous specifications into a bigger, more precise one which allows to share information easily. The categorical formalization of V-alignments allows for a rather simple description of the merge, and it is well-known that this can be described in terms of the category theoretic *pushout* construction. This topic has already been discussed in several places, and it is not the goal of this deliverable to repeat the respective argumentation (see [Kent, 2000], [Kalfoglou and Schorlemmer, 2002] or [Hitzler *et al.*, 2005b; Bouquet *et al.*, 2004a] for more details). However, we provide the following example.

Example 3. *The pushout of the V-alignment A from Example 2 is the triple $\langle M, g_1, g_2 \rangle$ with the merge M given by the following object:*

CURRENTLY UNAVAILABLE

The ontology refinements g_1 and g_2 are the obvious functions that are indicated by the chosen concept names.

[7]Note that ontologies that play the role of binary relations in V-alignments will usually not have any additional axioms.

CURRENTLY UNAVAILABLE

Figure 18.3: A Spanish ontology.

18.4.2 An algebra for simple alignments

The need for ontology alignment naturally arises when information from many ontologies is relevant to a given task. However, since the task of constructing alignments is not an easy one and can hardly be accomplished in a fully automatic fashion, it is reasonable to store and reuse known alignments. Any application like the Semantic Web will offer published ontologies as well as ontology alignments—sometimes partial, weak, or inconsistent—and there will be a need to integrate several alignments in a meaningful way. The purpose of this section is to introduce a sound algebra of V-alignments that allows for essential operations that enable us to compose, add, and intersect alignments.

Composing alignments

Composition is a central operation for the reuse of alignments: if we have an alignment between ontologies O_1 and O_2, and an alignment between O_2 and O_3, then it should be possible to obtain an alignment of O_1 and O_3. To formalize this operation, we employ a well-known categorical construction called *pullback*.

Definition 63 (Composition of alignments)**.** *Consider V-alignments $A = \langle |A|, p_1^A, p_2^A \rangle$ and $B = \langle |B|, p_2^B, p_3^B \rangle$ between ontologies O_1 and O_2, and O_2 and O_3, respectively. The* composition $B \circ A$ *of A and B is a V-alignment $C = \langle |C|, p_1^C, p_3^C \rangle$, defined as follows:*

– $|C| =_{\mathsf{def}} O$,

– $p_1^C =_{\mathsf{def}} p_1^A \circ f_A$,

– $p_3^C =_{\mathsf{def}} p_3^B \circ f_B$,

where $\langle O, f_A, f_B \rangle$ is the category-theoretic pullback of p_2^A and p_2^B.

This definition can be visualized by the following commutative diagram:

CURRENTLY UNAVAILABLE The V-alignment C is written in the same way as a composition of morphism: $C = B \circ A$.

For those knowledgeable in category theory this definition is quite obvious, but an intuition can also be given in a purely set-theoretical terms. Viewing V-alignments A and B as binary relations, the pullback C is just the well-known composition of relations (though care must be taken not to confuse our categorical $\circ$ with the symbol for relational composition, since the latter is usually read in a forward fashion). Though the pullback is closely related to composition of relations, it acts on (structured) ontologies instead of mere sets of pairs. Thus the pullback will also have ontological structure, which plays only a minor role in our investigations.

Example 4. *In order to give an example of composition of V-alignments, we need to extend our setting from Example 2 with a third ontology, shown in Figure 18.3. It models the same domain as before, but the labeling is in Spanish.*

CURRENTLY UNAVAILABLE

Figure 18.4: Intersection of alignments.

This ontology is aligned with O_2 according to V-alignment B that we sketch with the following relation: $\{$(Vehicle, Vehiculo), (Seaplane, Hydroavion)$\}$.

The expected composition alignment $B \circ A$ then is given by the relation $\{(Vehicule, Vehiculo), (Hydravion, Hydroavion)\}$.

Property 2. *Consider ontologies O_1, O_2, O_3, O_4 which are aligned by V-alignments A (O_1 and O_2), B (O_2 and O_3), and C (O_3 and O_4). The following properties hold:*

- *If $A \circ B = \langle X, f, g\rangle$ then $B \circ A = \langle X, g, f\rangle$. (Symmetry)*
- *$(C \circ B) \circ A = C \circ (B \circ A)$. (Associativity)*
- *$\langle |B \circ A|, f_A, f_B\rangle$ is a V-alignment for $|A|$ and $|B|$.*

The required proofs are straightforward. In our current setting, the above properties confirm that the proposed operation is well-behaved as a composition of alignments.

Intersection and union of alignments

Two V-alignments A and B for the same pair of ontologies O_1 and O_2 may give different information about the correspondences between O_1 and O_2. Given two V-alignments, one should be able to extract the consensual alignment, based on the agreed correspondences. This is called the *intersection* of alignments. Besides combining alignments in the appropriate way, there should exist an alignment which is more precise than both A and B and which gathers everything expressed in A and B. This new alignment should satisfy a kind of minimality property, i.e. it should not contain more information than that inside A or B. We call the result of this operation *union* of alignments, denoted by $A \cup B$.

These operations are indeed very useful in the context of the Semantic Web since they allow some kind of modularization of the alignments. In this respect, one can give a partial alignment with only a dozen or less correspondences and expect to retrieve more on the Web when needed.

Definition 64 (Intersection)**.** *Consider V-alignments $A = \langle |A|, f_1, f_2\rangle$ and $B = \langle |B|, g_1, g_2\rangle$ between ontologies O_1 and O_2. Let C be the limit of the diagram composed of alignments A and B (see Figure 18.4) associated to the morphisms $k_A : C \to A$, $k_B : C \to B$, $h_1 = f_1 \circ k_A$ and $h_2 = f_2 \circ k_A$. The* intersection *of A and B is a V-alignment $A \cap B = \langle C, h_1, h_2\rangle$.*

Union is defined via the intersection. In order to unify two alignments, one has to know what is common to both of them. Then the union is the disjoint union of this common part and the non-common parts. Formally, it is equivalent to a pushout of the intersection.

Definition 65 (Union of alignments)**.** *The union is the pushout of the intersection. Using the former notations, let $\langle C, i_A, i_B\rangle$ be the pushout of $\langle k_A, k_B\rangle$. The union of A and B is a V-alignment $\langle |A \cup B|, u_1, u_2\rangle$ such that $|A \cup B| = C$ and u_1 is the factorization of f_1 through i_A and u_2 is the factorization of f_2 through i_B.*

CURRENTLY UNAVAILABLE

Figure 18.5: Union of alignments.

CURRENTLY UNAVAILABLE

Figure 18.6: The expressivity of V-alignments is limited.

The definition is visualized in Figure 18.5.

The following example shows how union allows the definition of better alignments based on several weak or partial ones.

Example 5. *Let us assume there is a V-alignment C between O_1 and O_3 given by the relation $\{(\mathit{Volant},\ \mathit{Avion})\}$. Then, the intersection is just an empty alignment (i.e. alignment with an empty underlying ontology). Union is the alignment given by relation $\{(\mathit{Vehicle},\ \mathit{Vehiculo}), (\mathit{Seaplane},\ \mathit{Hydroavion}), (\mathit{Volant},\ \mathit{Avion})\}$.*

Intersection and union have the following properties:

Property 3. *Consider ontologies O_1 and O_2 and A, B, C V-alignments between them. The following properties hold:*

- $A \cap B = B \cap A$ *and* $A \cup B = B \cup A$ *(commutativity).*
- $(A \cap B) \cap C = A \cap (B \cap C)$ *and* $(A \cup B) \cup C = A \cup (B \cup C)$ *(associativity).*

Remark: In the general case, the properties of union and intersection do not coincide with those of set-theoretic union and intersection. For example, take the alignment E with underlying set $|E| = \{\mathit{Objet}\text{-}\mathit{Object}, \mathit{Objet}'\text{-}\mathit{Object}'\}$ and obvious associated projections. Then $E \cap E \neq E$. This is because E is a non-canonical representation of relation $\{(\mathit{Objet}, \mathit{Object})\}$.

18.4.3 The expressivity of simple alignments

In the previous section, we showed examples where the only relation existing between entities was equivalence of primitive concepts. In many other cases, the two ontologies to align are designed in such a way that a concept does not have its exact equivalent in the other ontology, although several concepts are closely related. For instance, one may find that concept $Woman$ in ontology O_1 is a subclass of concept $Person$ in ontology O_2. Let us look at this example more closely. In this case, the merge should obviously contain a concept $Person$ and a concept $Woman$ with a subsumption relation between them (see Figure 18.6). However, assuming this is the result of a pushout operation, it is not clear what the alignment should be.

Indeed, if morphisms are mere functions, then this simply cannot be a pushout. To the best of our knowledge, this problem has not really been investigated yet. Consequently, we have to consider other possible approaches to solve the problem:

1. Find more complex categories, where objects still are ontologies, but with morphisms able to express other relations;

2. Keep the category simple, and complexify the definition of an alignment using more elaborate structure;
3. Change the definition of the merge, for example by using a different type of colimit.

Although, it has not been established by formal proof, it seems not possible to design a concrete category of ontologies that would, at the same time, preserve the merge-as-pushout property and be able to express complex alignments (with subsumption and other non-symmetrical relations). Examples of categories of ontologies found in the literature are categories of theories or presentations in an institution [Goguen and Burstall, 1992]; or categories of local logic in the information flow theory [Barwise and Seligman, 1997]. They only offer the capability to express equivalence relations in our V-alignment framework. Their lack of expressivity with regard to V-alignments is a consequence of their being mere functions. However, by using relations or even sets of relations as morphisms, expressiveness increases at the cost of losing the merge-as-pushout principle.

However, these failed attempts do not prove the non-existence of a category in which most conceivable alignments are expressible as V-alignments. Indeed, if we consider ontology alignments as a (generalized) relation between valid interpretations of the ontologies, then one can define a highly non-practical category that is able to express any kind of alignments. Nonetheless, the lesson learned from the aforementioned investigations is that describing explicitly (syntactically) complex alignments with V-alignments is likely not feasible in practice. As a result, the second approach was examined and offered reasonable advantages among which the possibility to use already known concrete categories of ontologies. In the next section, a more complex structure, called W-alignment, is defined to improve on the expressivity of V-alignments. Under this extension, we also have to modify the formalization of merging, such that this approach also encompasses item (3) above.

18.5 Complex Alignments

As discussed in the previous section, the simplicity of V-shaped alignments comes at the price of a reduced expressivity of these constructions. In principle, they can only be used to formalize equivalences between syntactical expressions of the two ontologies.[8] In practical applications, ontologies can be related in much more complicated ways, that may involve complex, in general non-symmetrical relationships between their elements. In order to overcome this apparent restriction of our approach, this section introduces an extended formulation for alignments that we will suggestively dub *W-alignments*.

18.5.1 Bridge axioms for ontology alignment

Let us start with the simple example from Section 18.4.3: Consider two OWL-ontologies O_1 and O_2 that contain the atomic concepts $Woman$ and $Person$, respectively. Assuming that none of the ontologies contains both concepts, it is not possible to express the intended subsumption of $Woman$ and $Person$ with V-alignments.[9]

[8]To be more precise, it is possible to express complex alignments, but this will in general imply that pushout and merge will no longer coincide. The coincidence of pushout and merge, however, is a desirable property in many, though not all, ontology alignment manipulations.

[9]But note footnote 8.

CURRENTLY UNAVAILABLE

Figure 18.7: A W-alignment between two ontologies O_1 and O_2.

As a solution, one might consider introducing additional, possibly non-symmetric, types of relations between ontological concepts, thereby leaving the purely categorical formulation. This idea resembles the framework for alignments introduced in Deliverable 2.2.1 [Bouquet *et al.*, 2004a]. However, the obvious idea of extending correspondence relations between elements of an ontology (inside or outside a categorical setting) has strong limitations. While subsumption appears to be an important type of relationship, it is by far not the only one: disjointness of concepts, incomparability, or relationships between more than two concepts are examples of further relationships that are interesting in practice.

In effect, it is easy to see that there are infinitely many relevant relationships that could be considered in this context. For instance $\mathit{1} : \mathit{WW2}$, and instance of concept $\mathit{1} : \mathit{HistoricalEvent}$ in one ontology, may be related to $\mathit{2} : \mathit{TwentiethCentury}$, an instance of concept $\mathit{2} : \mathit{PeriodOfTime}$ in another ontology, via relation $\mathit{occursDuring}$. Explicitly introducing types for such relations seems not to be feasible, since the required relations depend on the structure of the alignment that one wants to express. We therefore prefer another well-known approach towards the formalization of ontological alignments, namely the introduction of *bridge axioms*.

Bridge axioms are arbitrary (onto)logical statements that describe the relationship between ontological concepts. A very simple case of bridge axiom for the above example is the statement "$\mathit{Woman} \sqsubseteq \mathit{Person}$" which expresses exactly the intended connection between the concepts. In general, this technique amounts to providing another set of ontological axioms, i.e. a third ontology, that describes how two ontologies are related. Clearly, this allows to formalize many types of relationships, but only within the borders of the ontology language under consideration. On the other hand, merging ontologies that are aligned in such a way becomes rather easy, since all connections are already available as expressions of the ontology language.

18.5.2 A categorical formulation of bridge axioms

It is not too complicated to cast the informal description of the previous section into a definition of more complex alignments along the lines of the categorical approach described earlier. The way to do this is to represent bridge axioms in form of an additional *bridge ontology*. The fact that certain concepts of the aligned ontologies occur within the bridge ontology is captured by V-alignments between the bridge ontology and each of the aligned ontologies. We thus arrive at the following definition:

Definition 66. *A* W-alignment *between two ontologies O_1 and O_2 consists of a* bridge ontology *B together with two V-alignments between O_1 and B and between O_2 and B, respectively.*

The situation is depicted in Figure 18.7, which also serves to illustrate why the above terminology was chosen. Note also that we do not impose any restrictions on the bridge ontology B. In particular B could contain axioms that are related to neither O_1 nor O_2. While this could be excluded by further specification, we shall continue working with the general definition above, and discuss possible restrictions later on.

CURRENTLY UNAVAILABLE

Figure 18.8: Merging with W-alignments.

CURRENTLY UNAVAILABLE

Figure 18.9: Computing the composition of W-alignments.

Based on this categorical formulation, it is now quite easy to give a suitable definition for merging of ontologies that are aligned with a W-alignment.

Definition 67. *Given two ontologies O_1 and O_2 and a W-alignment between them, the* merge *of O_1 and O_2 is defined to be the colimit of the respective alignment diagram.*

More explicitly, this colimit M is computed by successive pushouts as in Figure 18.8.

The ontologies O_1^+ and O_2^+ play an interesting role in ontology merging. Intuitively, they represent the original ontologies O_1 and O_2 extended with certain axioms and elements that enable us to express their alignment as a simple V-alignment. This idea is not entirely new, and elsewhere O_1^+ and O_2^+ have been called *portal ontologies*, referring to their specific role in making the knowledge of each of the ontologies accessible to the other one [Kent, 2000].

Due to our consistent categorical treatment of alignments, we can re-use earlier results to describe the merge in concrete cases. It is easy to see that merging in logic-based ontology languages like OWL can be achieved by just joining the axioms of both ontologies and the bridge ontology and identifying elements that are equivalent according to the V-alignments A_1 and A_2. Alternatively, one can compute the merge stepwise by constructing three pushouts, which will yield essentially the same result.

18.5.3 Composing W-alignments

Due to the increased complexity of W-alignments, a full-featured algebra along the lines of Section 18.4.2 would be more complicated than in the case of V-alignments. Nonetheless, we can easily describe a useful operation for composing W-alignments.

Definition 68. *Consider ontologies O_1, O_2, and O_3 with W-alignments as in Figure 18.9. The* composition *of the W-alignments of Figure 18.9 is described as follows:*

– *The bridge ontology B is obtained as the merge of the bridge ontologies B_1 and B_2, according to the W-alignment $\langle O_2, A_2, A_3 \rangle$,*

– *the V-alignment of O_1 and B is $\langle A_1, f_1, b_1 \circ g_1 \rangle$, and*

– *the V-alignment of O_3 and B is $\langle A_4, g_4, b_1 \circ f_4 \rangle$.*

The idea behind this definition is quite obvious: we know that there is a relation of O_1 and O_3, given by means of an intermediate ontology O_2. In order to describe this by a single bridge ontology, we just integrate both of the involved bridges with O_2. This construction has the advantage that it faithfully captures all information that is available about the composed alignment.

However, there is a major problem with the above definition: by deriving bridge axioms from the ontologies B_1, B_2, and O_2, we incorporate all the information in these knowledge bases into the new bridge ontology. But this set of bridge axioms might be highly redundant for the given purpose. For example, it may involve axioms of O_2 that are neither related to O_1 nor to O_3. Another pathological case is when O_2 is just the (disjoint) union of O_1 and O_3, while O_1 and O_3 are not related at all. In this case, we would rather wish the composed alignment to be empty, instead of containing the whole information of all involved ontologies.

Overcoming this difficulty relates to the problem of finding a minimal non-redundant set of axioms that yields a given set of desired (or relevant) conclusions. Unfortunately, logical languages tend to be highly non-local in this respect and a naïve syntactical approach of casting out irrelevant information is not feasible. One such overly simple idea would be to ignore bridge axioms that do not involve elements from the languages of both of the aligned ontologies. However, as the above example with concepts $Woman$ and $Person$ shows, it might be the case that meaningful bridge axioms relate elements exclusively to entities of the bridge ontology, which in turn carries the meaningful interrelation between its elements. Detecting whether some axiom eventually contributes to a relationship of elements from the aligned ontologies involves complex reasoning tasks, and we therefore do not attempt to provide a concise definition of a minimal bridge ontology here—this may be subject to further research.

18.6 Expressing DFOL Mappings

In this section, we describe how to model in our category-theoretic setting the formalisms specified in Chapter 17. In particular, we define a translation of the four interpretation constraints presented in Figure 17.1 into bridge axioms, thus proving a case in point that the W-alignments introduced above are indeed expressive enough to capture most of the previously designed approaches. Then a short informal discussion of soundness and completeness is given. The primary role of this section is to emphasize the generality of the categorical abstraction by embedding an example approach into our framework.

To achieve this goal, we will rely on a category of theories and theory morphisms as defined by institution theory [Goguen and Burstall, 1992]. In short, such a category has ontologies as objects, and its morphisms are just functions between the signatures (i.e. the vocabularies of the languages used to express the ontologies) of the ontologies. Additionally, these morphisms should preserve semantics, that is, they preserve semantical consequences between translated formulas. Since we are trying to model mappings in distributed first-order logic DFOL, we are only considering ontologies written in a fragment of first-order logic. So an object (ontology) O_i is a pair (L_i, A_i) where L_i is a first-order language with equality and A_i is a set of axioms in this language.

Now let us consider a set of interpretation constraints $\mathcal{IC}$ in DFOL, expressing an ontology alignment between O_1 and O_2. Its translation in the category-theoretic framework will be the W-alignment defined as follows (see Figure 3.6 for notation):

- O_1 and O_2 are the two ontologies to align, having signatures L_1 and L_2 respectively; for the sake of simplicity, we will assume $L_1 \cap L_2 = \emptyset$, which is not restrictive since renaming the elements of the vocabularies would result in an isomorphic language with the same semantics;

- B is the bridge ontology, written in the language $L_1 \cup L_2 \cup \{R\}$ with $R \notin L_1 \cup L_2$ being a binary predicate;

– A_1 and A_2 are ontologies with signatures L_1 and L_2, respectively, which do not contain any axiom;

– $f_1 : A_1 \to O_1, g_1 : A_1 \to B, f_2 : A_2 \to O_2, g_2 : A_2 \to B$ are ontology morphisms such that for each $e_i \in L_i$ we have that $f_i(e_i) = g_i(e_i) = e_i$.

The bridge ontology is built in the following way. For each constraint $c \in \mathcal{C}$, the following axioms are in B:

1. if c is of the form $i : P(x^{\to j}) \to j : Q(x)$ then add axiom $\forall x(P(x) \to \forall y(R(x, y) \to Q(y)))$;
2. if c is of the form $i : P(x) \to j : Q(x^{i\to})$ then add axiom $\forall x(P(x) \to \exists y(R(x, y) \wedge Q(y)))$;
3. if c is of the form $j : Q(x^{i\to}) \to i : P(x)$ then add axiom $\forall y(Q(y) \to \forall x(R(x, y) \to P(x)))$;
4. if c is of the form $j : Q(x) \to i : P(x^{\to j})$ then add axiom $\forall y(Q(y) \to \exists x(R(x, y) \wedge P(x)))$.

The axioms can be rewritten in a more standard way as follows:

1. $\forall x \forall y \neg P(x) \vee Q(y) \vee \neg R(x, y)$;
2. $\forall x \exists y \neg P(x) \vee Q(y) \wedge R(x, y)$;
3. $\forall y \forall x \neg Q(y) \vee P(x) \vee \neg R(x, y)$;
4. $\forall y \exists x \neg Q(y) \vee P(x) \wedge R(x, y)$.

18.6.1 Semantic considerations

The soundness and completeness of the translation just given depends on the semantics given to W-alignments. This semantics could be adequately described using e.g. institution theory, but it is not our goal to present this theory here. However, we will give an insight into the semantics by restricting ourselves to interpreting only W-alignments built with the translation given above. Moreover, in any case the semantics of the W-alignments relies on the underlying semantics of the language of the ontology. So we will use *interpretations* or *models* of an ontology to denote *local* interpretations or models.

In each interpretation $\mathcal{I}$ of B, the predicate R will be interpreted as a binary relation r_{12}. Moreover, since L_1 and L_2 are disjoint and do not contain the predicate R, the union $\mathcal{I} = m_1 \cup m_2 \cup \{r_{12}\}$ of interpretations m_1 and m_2 of L_1 and L_2, respectively, and the relation r_{12}, is an interpretation of the ontology B. Note that, $\langle\{m_1, m_2\}, \{r_{12}\}\rangle$ is a DFOL model of the distributed knowledge base $\{O_1, O_2\}$. With these notations, we have the following property:

Proposition 4. *$\langle\{m_1, m_2\}, \{r_{12}\}\rangle$ satisfies the interpretation constraints $\mathcal{IC}$ iff $\mathcal{I}$ is a (first-order) model of B. This generalizes easily to a set of local models $\mathcal{M}_1$ and $\mathcal{M}_2$.*

So the translation is at least complete. Additionally, if we restrict to this type of W-alignment, with interpretations $\mathcal{I}$ defined above, then the translation is also sound. A generalization is possible but we omit details here as this section aims at exemplifying the way concrete approaches can be embedded in our theoretical framework. Once such an embedding is done, the concrete approach automatically inherits the algebra as well as properties obtained in the abstraction.

Example 6. *We exemplify the previous definitions by translating the example given in Chapter 17, Section 17.1, where the following mappings are given:*

1. $1\colon Volant(x^{\rightarrow 2}) \rightarrow 2\colon Vehicle(x)$;
2. $1\colon Maison(x) \rightarrow 2\colon Domicile(x^{1\rightarrow})$;
3. $2\colon Seaplane(x^{1\rightarrow}) \rightarrow 1\colon Hydravion(x)$;
4. $2\colon Vehicle(x) \rightarrow 1\colon Vehicule(x^{\rightarrow 2})$.

We first need to work within a concrete category of ontologies. DFOL assumes ontologies are written in a fragment of first-order logic, for which institution theory [Goguen and Burstall, 1992] provides a category of first-order theories. In this theory, ontologies are pairs $\langle \Sigma, A\rangle$ where Σ is a signature, i.e. the symbols used to write the ontology (predicates, constants, functions, etc.), and A is a set of axioms. Simply said, ontology morphisms are signature morphisms such that the axioms of the first theory (ontology) are mapped to theorems of the second theory. So ontology O_1 signature is the set $\{Objet, Maison, Residence, Vehicule, Volant, Flottant, Hydravion\}$, and its axioms are $\forall x Residence(x) \rightarrow Maison(x)$, $\forall x Maison(x) \rightarrow Objet(x)$, etc. O_2 is defined in a similar way. We build the bridge ontology as follows: its signature is $\{Volant, Vehicle, Maison, Domicile, Seaplane, Hydravion, Vehicule, R\}$ and its axioms are $\forall x \forall y \neg Volant(x) \vee Vehicle(y) \vee \neg R(x,y)$, $\forall x \exists y \neg Maison(x) \vee Domicile(y) \wedge R(x,y)$, $\forall y \forall x \neg Seaplane(y) \vee Hydravion(x) \vee \neg R(x,y)$, $\forall y \exists x \neg Vehicle(y) \vee Vehicule(x) \wedge R(x,y)$. The two V-alignments just relate predicate symbols of B to those of O_1 and O_2 when they have the same name, with no additional axioms.

18.7 How to put our approach into practice

Let us now see how our approach can be used as a guidance for ontology merging. We noted earlier that the notion of product hinges only on (1) a decision regarding the ontology representation language used, and (2) the structural properties which shall be preserved by a morphism. The situation for pushouts is similar, we just need a third decision, namely (3) the fixing of an ontology mapping. Once these decisions have been made, the notion of pushout, and thus of the merging of two ontologies, is determined canonically, but one still needs to find a convenient concrete representation for the specified object (4). From there, conceptually sound algorithms for calculating both ontology mappings (5) and pushouts (6) can be devised.

Decisions need to be made step by step, and we propose the following workflow. Later steps, however, may indicate that earlier decisions need to be revised, and thus to retrace to earlier points.

1. *Decide on ontology representation language used.* This first step is probably the most unproblematic, since there are standard ontology languages around, and the specific application case will usually dictate the language. Potential candidates are e.g. F-Logic [Kifer

et al., 1995; Angele and Lausen, 2004] and different variants of OWL [Antoniou and van Harmelen, 2004].

2. *Determine what suitable morphisms are.* This step consists of describing the conditions which morphisms must satisfy. These conditions will primarily be dictated by the semantic interpretation of the ontology representation language chosen earlier, and by the specific requirements of the application case. Typical conditions could include the following.

   – The preservation of class hierarchies, i.e. functions shall be monotonic with respect to the *general class inclusion* orders on classes and/or roles.

   – The preservation of types (e.g. classes, roles, annotated objects).

   – The taking into account of model-theoretic logical properties, if featured by the underlying ontology representation language, like satisfiability, or the preservation of specific models.

   – The taking into account of proof-theoretic properties, i.e. such relating to particular inference methods chosen for reasoning with ontologies.

   – The preservation of language classes, e.g. by requiring that the merging of two OWL Lite ontologies shall not result in an OWL ontology which is not in OWL Lite.

3. *Determine what the ontology mapping is for this setting.* Usually, ontology mappings will be given by (binary) relations between elements of ontologies, indicating which elements shall be identified in the merging process. However, as the product of two ontologies may not always be described conveniently as a set of pairs of elements — as in the case of Set or Poset —, it needs to be understood at this stage, what the product really is, and thus what ontology mappings are in this setting.

4. *Determine what pushouts are for this setting.* While the characteristics of a pushout are fully determined by the previous steps, it is still necessary to find a particular instance of the pushout (both for the object and the embedding morphisms) in terms of the ontology language. This requires to define a possible result for arbitrary pushout operations and to show that it satisfies the formal requirements of a pushout. Difficulties at this stage arise from the fact that, like products, pushouts are not guaranteed to exist in general. Negative results may yield effective conditions for the existence of pushouts or even suggest a modification of the considered theory.

5. *Algorithmize how to obtain the mapping.* The issue of how to obtain suitable ontology mappings is a separate issue from the one discussed here, and will usually depend heavily on the application domain and on the ontology representation language chosen. Machine learning techniques may be used here together with linguistics-based approaches (see e.g. [Ehrig and Sure, 2004]). Fuzzy relations usually obtained by such approaches may however have to be defuzzified at some stage, in order to obtain a precise ontology mapping which will be used for the merging.

6. *Algorithmize how to obtain the pushout.* At this stage, it is theoretically clear what the pushout — and thus the merged ontology — will be. Casting this insight into an algorithm may require a considerable amount of work. The practitioner may also choose at this step to forego an exact implementation of the merging, and settle for an approximate or heuristic

approach for reasons of efficiency, while at the same time being guided by the exact merging result as the ontology to be approximated.

18.8 Conclusion

In this chapter, we have argued that the problem of merging ontologies based on a given ontology mapping can be formulated conveniently in the language of category theory. This leads to the well-known definition of the categorical pushout construction, which describes ontological merging independently from the concrete implementation that was chosen. Since pushouts do not exist in all categories, this also yields general guidelines for devising systems of interrelated ontologies.

Moreover, the definition helps circumscribing what are exactly the alignments: indeed, an alignment is not exactly any kind of relation as presented above. Since it should join correctly the two ontologies, then, there should be a viable push-out from this relation.

We also discussed general approaches for modelling alignments for a broad range of ontology languages. The basic concept of V-alignments is quite close in spirit to the alignment framework that was discussed previously, but it is more restricted in its expressivity. Building on these investigations, aligning with bridge axioms was formalized using W-alignments. This concept of alignment allows for global statements about the relationship between two ontologies, and is thus in a way more expressive than the framework based on binary relations. W-alignments appear to be a flexible and useful approach for the category-theoretical modelling, and we suggest that further investigations on category-theoretic aspects of alignment should build on these.

In general, the categorical viewpoint on alignments also allows to abstract from the details of a particular ontology language, and, in this general setting, to identify principle strengths of available alignment expression languages. In this sense, when further developed into a categorical framework for various concrete alignment formalisms, the presented approach can help potential users to have the required knowledge when choosing such a language.

On the other hand, it must be admitted that to-date non-local alignments with bridge axioms cannot easily be created in an automated fashion. The reason is that they express relationships that are very complex, and the search for such alignments imposes huge conceptional challenges. However, bridge axioms still have their use as an elegant and efficient formulation of alignments. Without introducing any meta-logical formalisms, they can already express complicated relations based on the underlying ontology language. This suggests them for use by human ontology engineers: anybody who is familiar with the given ontology language can also specify the relationships between the according ontologies easily. It thus might be beneficial to investigate ways of employing both paradigms for alignment in practical applications. Future research may furthermore yield efficient and practical algorithms for automated or semi-automated complex alignments based on W-alignments.

Methods and insights from category theory could be used to assist in the development both of rigorous theoretical settings for ontology merging and of conceptually sound algorithms for practical implementations. In addition it yields a direct definition of the merge of two ontologies once an alignment is provided. Conversely, similar considerations can also be useful to validate alignment and merging constructions that have been conceived exclusively on practical grounds, since one may ask in which sense (in which category) a given merging process produces results of general validity.

Part IV

Evaluation

Chapter 19

Introduction: purpose, method and types of do2002a for ontology alignment

Aligning ontologies consists of finding the corresponding entities in these ontologies. There have been many different techniques proposed for implementing this process (see deliverable 2.2.3). They can be classified along the many features that can be found in ontologies (labels, structures, instances, semantics), or with regard to the kind of disciplines they belong to (e.g., statistics, combinatorics, semantics, linguistics, machine learning, or data analysis) [Rahm and Bernstein, 2001; Kalfoglou and Schorlemmer, 2003b; Euzenat *et al.*, 2004a]. The alignment itself is obtained by combining these techniques towards a particular goal (obtaining an alignment with particular features, optimising some criterion). Several combination techniques are also used. The increasing number of methods available for schema matching/ontology integration suggests the need to establish a consensus for do2002a of these methods.

Beside this apparent heterogeneity, it seems sensible to characterise an alignment as a set of pairs expressing the correspondences between two ontologies. We proposed, in deliverable 2.2.1, to characterise an alignment as a set of pair of entities (e and e'), coming from each ontologies (o and o'), related by a particular relation (R). To this, many algorithms add some confidence measure (n) in the fact the relation holds [Euzenat, 2003; Bouquet *et al.*, 2004a; Euzenat, 2004].

From this characterisation it is possible to ask any alignment method, given

- two ontologies to be aligned;
- an input partial alignment (possibly empty);
- a characterization of the wanted alignment (1:+, ?:?, etc.).

to output an alignment. From this output, the quality of the alignment process could be assessed with the help of some measurement. However, very few experimental comparison of algorithms are available. It is thus one of the objectives of Knowledge web and other people worldwide to run such an do2002a. We have participated in the organisation of two events in 2004 which are the premises of a larger do2002a event:

- The Information Interpretation and Integration Conference (I3CON), held at the NIST Performance Metrics for Intelligent Systems (PerMIS) Workshop, is an ontology alignment demonstration competition on the model of the NIST Text Retrieval Conference. This contest has focused ”real-life” test cases and comparison of algorithm global performance.

– The Ontology Alignment Contest at the 3rd Evaluation of Ontology-based Tools (EON) Workshop, held at the International Semantic Web Conference (ISWC), targeted the characterisation of alignment methods with regard to particular ontology features. This contest defined a proper set of benchmark tests for assessing feature-related behavior.

These two events are described more thoroughly in Appendix A.

Since all benchmarking activity must be carried out with a systematic procedure on clearly defined tasks. This is the purpose of this deliverable to propose such a procedure. This introduction will define the general objective of evaluating the alignment algorithms, the kind of tests which can be performed and the overall methodology to be followed.

19.1 Goal of do2002a

As mentionned in deliverable D2.1.1, do2002a should enable the measure of the degree of achievement of proposed tasks on a scale common to all methods. The main features of benchmarking are:

– measurement via comparison;
– continuous improvement;
– systematic procedure.

A benchmark is a test that measures the performances of a system or subsystem on a well defined task or set of tasks (comp.benchmark.FAQ). In fact, the two first items are not really the same goal and we will identify different types of do2002a later on.

The major and long term purpose of the do2002a of ontology alignment methods is to help designers and developers of such methods to improve them and to help users to evaluate the suitability of proposed methods to their needs. The benchmarking considered here should help research on ontology alignment. For that purpose, the do2002a should help evaluating absolute performances (e.g., compliance) and relative performances (e.g., in speed or accuracy).

The medium term goal is to set up a set of reference benchmark tests for assessing the strengths and weaknesses of the available tools and to compare them. Some of these tests are focussing the characterisation of the behaviour of the tools rather than having them compete on real-life problems. It is expected that they could be improved and adopted by the algorithm implementers in order to situate their algorithms. Building benchmark suites is highly valuable not just for the group of people that participates in the contests, but for all the research community. The do2002a should thus be run over several years in order to allow the measure of the evolution of the field.

The shorter term goal of the initiatives launched in 2004 was firstly to illustrate how it is possible to evaluate ontology alignment tools and to show that it was possible to build such an do2002a campaign. It is a common subgoal of do2002a campaign that their return helps improving the do2002a methodologies.

19.2 Types of do2002as

There can be several classifications of benchmarks depending on the criteria used. We can divide benchmarking with regard to what they are supposed to evaluate:

competence benchmarks allows to characterise the level of competence and performance of a particular system with regard to a set of well defined tasks. Usually, tasks are designed to isolate particular characteristics. This kind of benchmarking is relevant to kernel benchmark or unit tests;

comparison benchmark allows to compare the performance of various systems on a clearly defined task or application.

The goal of these two kinds of benchmarks are different: competence benchmarks aim at helping system designers to evaluate their systems and to localise them which regard with a common stable framework. It is helpful for improving individual systems. The comparison benchmarks enables to compare systems with regard to each others on a general purpose tasks. Its goal is mainly to help improving the field as a whole rather than individual systems. These two kinds of benchmarks are futher considered below.

In deliverable D2.1.4, the following classification, due to [Stefani *et al.*, 2003], describes the four following types of benchmarks that can be used in the do2002a of software systems:

Application benchmarks These benchmarks use real applications and workload conditions.

Synthetic benchmarks These benchmarks emulate the functionalities of significant applications, while cutting out additional or less important features.

Kernel benchmarks These benchmarks use simple functions designed to represent key portions of real applications.

Technology-specific benchmarks These benchmarks are designed to point out the main differences of devices belonging to the same technological family.

Each of these approaches have advantages and drawbacks. We will see that the I3CON experiment choose the first approach and ended with the second, while the EON initiative has used the fourth option.

This classification is concerned by the way to design benchmarks while the competence /performance classification is based on what is evaluated by the benchmarks. These two are not totally independent as the phrasing suggests it. Since we are first interested by the "what to evaluate" rather than the "how", we will focus on competence/performance.

19.2.1 Competence benchmark

Competence benchmarks aim at characterising the kind of task each method is good at. There are many different areas in which methods can be evaluated. One of them is the kind of features they use for finding matching entities (this complements the taxonomy provided in [Rahm and Bernstein, 2001]):

terminological (T) comparing the labels of the entities trying to find those which have similar names;

internal structure comparison (I) comparing the internal structure of entities (e.g., the value range or cardinality of their attributes);

external structure comparison (S) comparing the relations of the entities with other entities;

extensional comparison (E) comparing the known extension of entities, i.e. the set of other entities that are attached to them (in general instances of classes);

semantic comparison (M) comparing the interpretations (or more exactly the models satisfying the entities).

A set of reference benchmarks, targetting one type of feature at a time can be defined. These benchmarks would caracterize the competence of the method for one of these particular features of the languages.

19.2.2 Performance benchmarks: competition

Performance benchmarks are aimed at evaluating the overall behaviour of alignment methods in versatile real-life examples. It can be organised as a yearly or bi-annual challenge (la TREC) for comparing the best compound methods. Such benchmarks should yield as a result the distance between provided output and expected result as well as traditional measures of the amount of resource consumed (time, memory, user input, etc.).

19.3 Evaluation methodology

Each do2002a must be carried out according to some methodology. Knowledge web deliverable D2.1.4 presents a benchmarking methodology that is briefly summarized here.

The benchmarking process defined in the methodology is a continuous process that should be performed indefinitely in order to obtain a continuous improvement both in the tools and in the same benchmarking process. This process is composed of a benchmarking iteration that is repeated forever and that is composed of three phases (Plan, Experiment, and Improve) and ends with a Recalibration task.

The three phases of each iteration are the following:

Plan phase It is composed of the set of tasks that must be performed for indentifying the goal and the subject of the do2002a (see above), preparing the proposal for benchmarking, finding other organisations that want to participate in the benchmarking activity (Section 2.2), and planning the benchmarking.

Experiment phase It is composed of the set of tasks where the experimentation over the different tools that are considered in the benchmarking activity is performed. This includes defining the experiment and its tool set, processing and analysing the data obtained, and reporting the experimentation results.

Improve phase It is composed of the set of tasks where the results of the benchmarking process are produced and communicated to the benchmarking partners, and the improvement of the different tools is performed in several improvement cycles. Precising how to report and communicate on the results is considered in §24; while planning the corrective methods, improving the actual systems and monitoring the results is a matter concerning algorithms developers and is not covered in this deliverable.

While the three phases mentioned before are devoted to the tool improvement, the goal of the **Recalibration** task is to improve the benchmarking process itself after each benchmarking iteration, using the lessons learnt while performing the benchmarking.

We are, in this deliverable, mainly concerned with the design of the do2002a, i.e., the Plan and Experiment phases described above. The processing and recalibrating of the do2002a is, in theory the topic of deliverable D2.2.4. However, since, we already run two do2002a events in 2004, we have implemented these two steps already and this deliverable can be seen as the end of the recalibrating phase. The two experiments are reported in appendix. The remainder of this deliverable will consist in proposing mainly the main outline for the Plan phase. The Experiment phase will be the subject of deliverable 2.2.4.

19.4 Conclusion

The goal of the Knowledge web evalution effort is the improvement of ontology alignment techniques. For that purpose we will define the kind of tests to be processed and measures for assessing the results. This will be done for two kinds of tests: competence and performance benchmarks.

Next chapter evaluates what is the variability in the alignment task, and, consequently, what are the parameters that must be controlled in its do2002a. Chapter 21 considers the potential do2002a metrics that can be used in order to assess the performance of the evaluated algorithms. Chapter 24 provides the definition of a possible do2002a process, including the identification of actors and Chapter 26 describes the kind of support which is provided to the community in order to perform these do2002as. The last chapter will provide some guidelines for defining benchmark tests in order to evaluate alignments.

Chapter 20

Dimensions and variability of alignment do2002a

The goal of this chapter is to characterize the variability of the alignment task in order to assess the limitations of the benchmark tests or to design benchmarks spanning the whole spectrum of alignment and to know what variable must be controlled during their design.

Deliverable 2.2.1 provided a precise definition of the alignment process which is recalled here. The alignment process simply consists of generating an alignment (A') from a pair of ontologies (o and o'). However, there are various other parameters which can extend the definition of the alignment process. These are namely, the use of an input alignment (A) which is to be completed by the process, the alignment methods parameters (which can be weigths for instance) and some external resources used by the alignment process (which can be general-purpose resources not made for the case under consideration, e.g., lexicons, databases). This process can be defined as follow:

Definition 69 (Alignment process)**.** *The alignment process can be seen as a function f which, from a pair of ontologies o and o' to align, an input alignment A, a set of parameters p, a set oracles and resources r, returns a new alignment A' between these ontologies:*

$$A' = f(o, o', A, p, r)$$

This can be represented as in Figure 20.1.

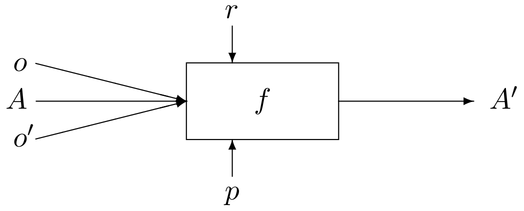


Figure 20.1: The alignment process.

Each of the elements featured in this definition can have specific characteristics which influence the difficulty of the alignment task. It is thus necessary to know and control these characteristics (called dimensions because they define a space of possible tests). The purpose of the dimensions is the definition of the parameters and characteristics of expected behavior in benchmark. Indeed, for each dimension a specific benchmark could be designed. However, there are too many of them and it is thus necessary to choose fixed values for most of these possible parameters.

We review below all the dimensions and justify some choices in designing benchmarks.

20.1 Input ontologies

Input ontologies (o, o') can be characterised by three different dimensions:

Heterogeneity of the input languages: are they described in the same knowledge representation languages? This corresponds to asking for the non emptyness of the syntactic component of the resulting alignment.

Languages: what are the languages of the ontologies? Example of languages are KIF, OWL, RDFS, UML, F-Logic, etc. as well as variant of these e.g., OWL-Lite, OWL-DL, OWL-Full.

Number: is this an alignment or a multi-alignment?

Currently, Knowledge web considers the alignment of ontologies expressed in the same language. The rationale for this is that language translation or language mapping resort to very specific techniques different from those used for aligning two ontologies. These techniques can be set up independently of any ontology. We thus consider that when confronted with ontologies expressed in different languages, it is better to first translate one of the ontology into the language of the other before processing an alignment properly speaking.

All the languages mentioned above are worth considering. However, in the setting up of a particular test, it is necessary to decide for the use of one language. During the first meetings of the 2.2 work package, it has been considered that the OWL language was the choice to consider first. Moreover, we decided for the OWL-DL fragment of OWL. During the first campaign we run, some of the competitors first translated the test from OWL to RDFS before running their algorithms. It is perfectly admissible that not all the benchmark campaign use the same languages.

Tasks involving multi-alignment are very specific. Indeed, usually alignment is triggered by editors that want to expand an ontology or web services to compose. This involves the alignment of two ontologies. Bringing other ontologies in the process does not help solving the problem. Multi-alignment is rather reserved to ontology normalisation or mining. For the moment it seems preferable to consider only two ontologies to align. This should hold until competitors complain that multi-alignment would be worthwhile.

20.2 Input alignment

The input alignment (A) can have the following characteristics:

Complete/update: Is the alignment process required to complete an existing alignment? (i.e., is A non empty).

Multiplicity : How many entities of one ontology can correspond to one entity of the others? (see “Output alignment”).

For the first kind of benchmark it seems reasonable that no input alignment will be provided. Of course, the competitors are free to design their method around the composition of various methods which provide intermediate alignments.

20.3 Parameters

Parameters (p, r) of the alignment process were identified as:

Oracles/resources Are oracle authorized? If so, which ones (the answer can be any)? Is human input authorized?

Training Can training be performed on a sample?

Proper parameters Are some parameter necessary? And what are they? This point is quite important when a method is very sensitive the variation of parameters. A good tuning of these must be available.

Many systems take advantage of some external resources such as WordNet, sets of morphological rules or a previous alignment of general purpose catalogues (Yahoo and Google for instance). It is perfectly possible to use these resources as long as they have not been tuned to the task for the current benchmark (for instance, using a sub-lexicon which is dedicated to the domain considered by the tests). Of course, it is perfectly acceptable that the algorithms prune or adapt these resources to the actual ontologies. This is considered as the normal process of the algorithm. However this processing time must be considered within the running time of the algorithm.

Some algorithms could take advantage of the web for selecting some resource that is adapted to the considered ontology. This is perfect behaviour. However, as long as this is not specifically required by some competitor and because this is quite difficult to control, we think that this should not be authorised in the first place.

In general, if human input is provided, the performance of systems can be expected to be better. However, in the current state, which is the absence of any consensus or valuable methods for handling and evaluating the contribution of this human input, we will not take this into account.

Training on some sample is very often used by methods for aligning ontologies and mapping schemas. However, this training sample is a particular alignment. The only situation in which this makes a lot of sense is when a user provides some example of aligned instances and the system can induce the alignment from this. This is thus quite related to user input. We consider that this is an interesting characteristics to be considered in a second step.

Of course, some parameters can be provided to the methods participating in the do2002a. However, these parameters must be the same for all tests. It can be the case that some methods are able to tune their parameters depending on the presented ontologies. In such a case, the tuning process is considered part of the method. However, this process must be computed from the ontology input only, not from externally provided expected results.

It seems necessary, in competence benchmark, to have participants providing the best parameter set they found for the benchmark. This set must be the same for all tests. In competitive tests, especially when the expected result is not known from the participants, they will not change their parameters.

20.4 Output alignment

We identify the following possible constraints on the output alignment (A') of the algorithm:

Multiplicity How many entities of one ontology can correspond to one entity of the others? Usual notations are 1:1, 1:m, n:1 or n:m. We prefer to note if the mapping is injective, surjective and total or partial on both side. We then end up with more alignment arities (noted with, 1 for injective and total, ? for injective, + for total and * for none and each sign concerning one mapping and its converse): ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:*. These assertions could be provided as input (or constraint) for the alignment algorithm or be provided as a result by the same algorithm.

Justification Is a justification of the results provided?

Relations Should the relations involved in the correspondences be only equivalence relations or could they be more complex?

Strictness Can the result be expressed with trust-degrees different than $\top$ and $\bot$ or should they be strictified before?

In real life, there is no reason why two independently developed ontologies should have a particular alignment multiplicity other than *:*. This should be the (non) constraint on the output alignment of the benchmark tests. However, if we say so and all our tests provides some particular type of alignment (for instance, ?:? in the EON ontology tests), it can be said that this introduces a biais. This biais can be suppressed by having each type of alignment equally represented. However, this is not easy to find and this is not realistic. What would be realistic would be to have a statistical do2002a of the proportion of each type of alignment. In the absence of such an do2002a, however, it remains reasonable to stick to the *:* rule. This could be revised later on.

Another worthwhile feature for users is the availability of meaningful explanations or justifications of the correspondences. However, very few algorithms are able to deliver them and there is no consensus either on the form in which they are expressed neither on the way to compare them. So, it is currently not possible to ask for explanations in the benchmark results.

As mentioned in Deliverable 2.2.1 and 2.2.3, all algorithms deliver pairs of entities (called correspondences). However, some of them associate a relation between the entities different from equivalence (e.g., specificity) and some of them associate a strength to the correspondence (which can be a probability measure). A problem is that not all algorithms deliver the same structure. Moreover, alignments must be used in tasks for which, most of the time it is necessary to know how to interpret a term of one ontology with regard to another ontology. For these reasons, and because each method can, at least, deliver equivalence statement with the maximum strength, it seems better to avoid using any kind of relation or measure (more exactly, to design the tests with alignment involving only equivalence relations and $\top$ confidence measure.

20.5 Alignment process

The alignment process (f) itself can be constrained by:

Resource constraints Is there a maximal amount of time or space available for computing the alignment?

Language restrictions Is the mapping scope limited to some kind of entities (e.g., only T-box, only classes)?

Property Must some property be true of the alignment? For instance, one might want that the alignment (as defined in the previous chapter be a conseqeunce of the combination of the ontologies (i.e., $o, o' \models A'$) or that alignments preserve consequences (e.g., $\forall \phi, \phi' \in L, \phi \models \phi' \implies A'(\phi) \models A'(\phi')$) or that the initial alignment is preserved (i.e., $o, o', A' \models A$).

Resource constraints can be considered either as a constraint (the amount of resource is limited) or a result (the amount consumed is measured – see Chapter 21). It is a relatively important factor, at least for performance tests and must be measured. This can also be measured for competence tests (even if it is absolutely difficult to do because of the heterogeneity of the environments in which these algorithms can be run).

Constraints on the kind of language construct to be found in mappings can be designed. However, currently very few alignment algorithms can align complex expressions, most of them align the identified (named) entities and some of them are only restricted to concepts. With regard to its importance and its coverage by current alignment systems, it makes sense to ask for the alignment of named entities and consider complex expressions later.

The properties of the alignments provided by the alignment algorithms are not very often mentioned and they seems to be very heterogeneous depending of the implemented techniques. It seems thus difficult to ask for particular properties. As for the type of alignment, not asking for a property is a problem if the tests do not satisfy a variety of properties. Moreover, it is not obvious that in real life, there are any properties to be satisfied by alignments (because ontologies are made for different purposes). So, at this stage, we do not commit to a particular property.

20.6 Conclusion

We propose to focus first on the simplest kind of test:

- comparing *two* ontologies written in the *same language*: OWL-DL;
- without input alignment;
- with any kind of fixed parameters and any kind of fixed and general purpose resources;
- without any kind of user input nor training samples;
- provide a strict *:* equivalence alignment of named entities;
- and measure the amount of resources consumed.

Like TREC has evolved towards multi-track competitions considering different benchmark set-up, it seems reasonable that the decision proposed here will have to be reconsidered with the evolution of the field.

It will then be natural to have extensions around the following features (ordered by perceived importance):

- considering another language than OWL;
- considering any kind of external resources (use of the web as it is);
- considering non-strict alignments and alignments with various types of relations;
- considering aligning with complex kind of expressions.

or specific tracks around (ordered by perceived importance):

– alignment with training on some sample seems a very important task;
– alignment with human input;
– alignment under difficult resource constraints (and even anytime alignment);
– alignments satisfying some formal properties;
– considering the alignment completion task;
– depending on task, consider more specific types of alignments (e.g., 1:1).

Chapter 21

Evaluation measures

This chapter is concerned with the question of how to measure the do2002a results returned by benchmarking. It considers a wide range of different possible measures for evaluating alignment algorithms and systems. They include both qualitative and quantitative measures. We divide them into compliance measures which evaluate the degree of conformance of the alignment methods to what is expected, performance measures which measure non functional but important features of the algorithms (such as speed), user-related measures focusing on user do2002a, overall aggregating measures, and measures to evaluate specific tasks or applications.

21.1 Compliance measures

Compliance measures evaluate the degree of compliance of a system with regard to some standard. They can be used for computing the quality of the output provided by a system compared to a reference output. Note that such a reference output is not always available, not always useful and not always consensual. However, for the purpose of benchmarking, we can assume that it is desirable to provide such a reference.

21.1.1 Precision, recall, and others

There are many ways to qualitatively evaluate returned results [Do *et al.*, 2002]. One possibility consists of proposing a reference alignment (R) that is the one that the participants must find (a *gold standard*). The result from the evaluated alignment algorithm (A) can then be compared to that reference alignment. In what follows, the alignments A and R are considered to be sets of pairs.

The most commonly used and understood measures are precision (true positive/retrieved) and recall (true positive/expected) which have been adopted for ontology alignment. They are commonplace measures in information retrieval.

Definition 70 (Precision). *Given a reference alignment R, the precision of some alignment A is given by*

$$P(A,R) = \frac{|R \cap A|}{|A|}.$$

Please note, that precision can also be determined without explicitly having a complete reference alignment. Only the correct alignments among the retrieved alignments have to be determined ($R \cap A$), thus making this measure a valid possibility for ex-post do2002as.

Definition 71 (Recall)**.** *Given a reference alignment R, the recall of some alignment A is given by*

$$R(A,R) = \frac{|R \cap A|}{|R|}.$$

The fallout measures the percentage of retrieved pairs which are false positive.

Definition 72 (Fallout)**.** *Given a reference alignment R, the fallout of some alignment A is given by*

$$F(A,R) = \frac{|A| - |A \cap R|}{|A|} = \frac{|A \setminus R|}{|A|}.$$

Precision and recall are the most widely and commonly used measures. But usually, when comparing systems one prefers to have only one measure. Unfortunately, systems are often not comparable based solely on precision and recall. The one which has higher recall has lower precision and vice versa. For this purpose, two measures are introduced which aggregate precision and recall.

The F-measure is used in order to aggregate the result of precision and recall.

Definition 73 (F-measure)**.** *Given a reference alignment R and a number α between 0 and 1, the F-measure of some alignment A is given by*

$$M_\alpha(A,R) = \frac{P(A,R) \cdot R(A,R)}{(1-\alpha) \cdot P(A,R) + \alpha \cdot R(A,R)}.$$

If $\alpha = 1$, then the F-measure is equal to precision and if $\alpha = 0$, the F-measure is equal to recall. In between, the higher α, the more importance is given to precision with regard to recall. Very often, the value $\alpha = 0.5$ is used, i.e. $M_{0.5}(A,R) = \frac{2 \times P(A,R) \times R(A,R)}{P(A,R)+R(A,R)}$, the harmonic mean of precision and recall.

The overall measure (defined in [Melnik *et al.*, 2002] as accuracy) is an attempt of measuring the effort required to fix the given alignment (the ratio of the number of errors on the size of the expected alignment). Overall is always lower than the F-measure.

Definition 74 (Overall)**.** *Given a reference alignment R, the overall of some alignment A is given by*

$$O(A,R) = R(A,R) \times \left(2 - \frac{1}{P(A,R)}\right).$$

It can also be defined as:

$$O(A,R) = \frac{|(A \cup R) - (A \cap R)|}{|R|}.$$

When comparing systems in which precision and recall can be continuously determined, it is more convenient to draw the precision/recall curve and compare these curves. This kind of measure is widespread in the results of the TREC competitions.

21.1.2 Weighted Hamming distance

The Hamming distance measures the similarity between two alignments by counting the joint correspondences with regard to the correspondence of both sets.

Definition 75 (Hamming distance)**.** *Given a reference alignment R, the Hamming distance between R and some alignment A is given by*

$$H(A, R) = 1 - \frac{|A \cap R|}{|A \cup R|}.$$

The Weighted Hamming distance pays attention not only to the correspondences but to their strengths as well. It requires that the strengths (as defined in deliverable D2.2.1) be the same in both sets of correspondences.

Definition 76 (Weighted Hamming distance)**.** *Given a reference alignment R, the weighted Hamming distance between R and some alignment A is given by*

$$W(A, R) = \sum_{c \in A \cup R} \frac{|strength_A(c) - strength_R(c)|}{|A \cup R|}$$

in which $strength_X(c)$ is 0 *if $c \notin X$.*

However, since the semantics of strength is not well defined, it is hazardous to use them for comparing alignments. Moreover, it can be considered that some reference alignment is always achievable in each context. In such a case, it would be useful to compare an exact (hardened) version of each obtained alignment rather than a rough alignment unless the way it is used is known.

It could be more interesting to measure from how far the alignment missed the target. To that extent it would be necessary to measure a distance from an obtained alignment and a reference alignment. However, this distance seems currently tricky to define for several reasons:

– it shall highly depend on the task to be performed;
– it will introduce a biais in the do2002a of the algorithms in favour of those based on this distance. This is not acceptable unless it is certain that this distance is the best one.

21.2 Performance measures

Performance measures (or non-functional measures) measure the resource consumption for aligning two ontologies. They can be used when the algorithms are 100% compliant or balanced against compliance [Ehrig and Staab, 2004]. Unlike the compliance measures, performance measures depend on the benchmark processing environment and the underlying ontology management system. Thus it is rather difficult to obtain objective do2002as.

21.2.1 Speed

Speed is measured in amount of time taken by the algorithms for performing their alignment tasks. If user interaction is required, one has to ensure to effectively measure the processing time of the machine only.

21.2.2 Memory

The amount of memory used for performing the alignment task marks another performance measure. Due to the dependency with underlying systems, it could also make sense to measure only the extra memory required in addition to that of the ontology management system (but it still remain highly dependent).

21.2.3 Scalability

There are two possibilities for measuring scalability, at least in terms of speed and memory requirements. First, it can be assessed by theoretical study. And second, it can be assessed by benchmark campaigns with quantified increasingly complex tests. From the results, the relationship between the complexity of the test and the required amount of resources can be represented graphically and the mathematical relationship can be approximated.

21.3 User-related measures

So far the measures have been machine focused. In some cases algorithms or applications require some kind of user interaction. This can range from the user utilizing the alignment results to concrete user input during the alignment process. In this case, it is even more difficult to obtain some objective do2002a. This subsection proposes measures to get the user into the do2002a loop.

21.3.1 Level of user input effort

In case algorithms require user intervention, this intervention could be measured in terms of some elementary information the users provide to the system. When comparing systems which require different input or no input from the user, it will be necessary to consider a standard for elementary information to be measured. This is not an easy task.

21.3.2 General subjective satisfaction

From a use case point of view it makes sense to directly measure the user satisfaction. As this is a subjective measure it cannot be assessed easily. Extensive preparations have to be made to ensure a valid do2002a. Almost all of the objective measures mentioned so far have a subjective counterpart. Possible measurements would be:

– input effort,
– speed,
– resource consumption (memory),
– output exactness (related to precision),
– output completeness (related to recall),
– and understandability of results (oracle or explanations).

Due to its subjective nature numerical ranges as do2002a result are less appropriate than qualitative values such as very good, good, satisfactory, etc.

21.4 Aggregated measure

Different measures suit different do2002a goals. If we want to improve our system, it is best to have as many indicators as possible. But if we want to single out the best system, it is generally easier to evaluate with very few or only one indicator. To allow for this, the different individual measurements have to be aggregated. This can be achieved by giving every measurement a weight (e.g., in form of a weighted linear aggregation function). Obviously the weights have to be chosen carefully, again dependent on the goal.

Definition 77 (Aggregated measure)**.** *Given a set of do2002a measures $m_i \in M$ and their weighting $w_i \in W$, the aggregated measure $Aggr$ is given by*

$$Aggr(M, W) = \sum_{m_i \in M} w_i \cdot m_i.$$

21.5 Task specific do2002a

So far do2002a was considered in general. But the do2002a could also be considered in the context of a particular task.

As a matter of fact, there are tasks which require high recall (for instance aligning as a first step of an interactive merge process) and others which require high precision (e.g. automatic alignment for autonomously connecting two web services). Different *task profiles* could be established to explicitly compare alignment algorithms with respect for certain tasks. The following short list of possible scenarios gives hints on such scenarios (taken deliverable 2.2.3):

– Agent communication,
– Emergent semantics,
– Web service integration,
– Data integration,
– Information sharing and retrieval from heterogeneous sources,
– Schema alignment or merging in overlay networks.

In terms of measurements, it would be useful to set up experiments which do not stop at the delivery of alignments but carry on with the particular task. This is especially true when there is a clear measure of the success of the overall task. Even without this, it could be useful to share corresponding aggregate measures associated to these "task profile".

Nevertheless, it will be extremely difficult to determine the do2002a value of the alignment process independently. The effects of other components of the overall application have to carefully filtered out.

21.6 Conclusion

This chapter presented several approaches to measure do2002as ranging from quality to resource consumption, from machine-focused to user-focused, and from general to task-specific measures.

However, it seems that currently the most natural factors to measure quality are precision and recall because they can be interpreted easily.

The next kind of measure to consider in upcoming benchmarking efforts are resource consumption and task-specific do2002as. Despite the different kinds of problems for the do2002a, which have to be overcome first, these measures are important for reaching the next steps of ontology alignment algorithms and should therefore be considered in very near future.

Chapter 22

Generalised measures

22.1 Introduction

In order to evaluate the performance of matching algorithms it is necessary to confront them with ontologies to match and to compare the results based on some criterion. The most prominent criteria are precision and recall originating from information retrieval and adapted to the matching task. Precision and recall are based on the comparison of the resulting alignment A with another standard alignment R, effectively comparing which correspondences are found and which are not. These criteria are well understood and widely accepted.

However, as we have experienced in last year's Ontology Alignment Contest [Sure *et al.*, 2004], they have the drawback to be of the all-or-nothing kind. An alignment may be very close to the expected result and another quite remote from it and both return the same precision and recall. The reason for this is that the criteria only compare two sets of correspondences without considering if these are close or remote to each other: if they are not the same exact correspondences, they score zero. They both score identically low, despite their different quality. It may be helpful for users to know whether the found alignments are close to the expected one and easily repairable or not. It is thus necessary to measure the proximity between alignments instead of their strict equality.

In this chapter we investigate some measures that generalize precision and recall in order to overcome the problems presented above. We reproduce here the main part of [Ehrig *et al.*, 2005]. We first provide the basic definitions of alignments, precision and recall as well as a motivating example (§22.2). We then present a framework for generalizing precision and recall (§22.3). This framework is instantiated by four different measures (including classical precision and recall) (§22.4) and we show on the motivating example that the proposed measures do not exhibit the rigidity of classical precision and recall (§22.5).

22.2 Foundations

22.2.1 Alignment

We consider the result of matching, called alignment, as a set of pairs of entities $\langle e, e' \rangle$ from two ontologies O and O' that are supposed to satisfy a certain relation r with a certain confidence n.

Definition 78 (Alignment, correspondence)**.** *Given two ontologies O and O', an alignment between O and O' is a set of correspondences (i.e., 4-uples): $\langle e, e', r, n \rangle$ with $e \in O$ and $e' \in O'$ being the two matched entities, r being a relationship holding between e and e', and n expressing the level of confidence $[0..1]$ in this correspondence.*

A matching algorithm returns an alignment A which is compared with a reference alignment R. Let us illustrate this through a simple example. Figure 22.1 presents two ontologies together with two alignments A_1 and R. In this example, for the sake of simplification, the relation is always '=' and the confidence is always 1.0.

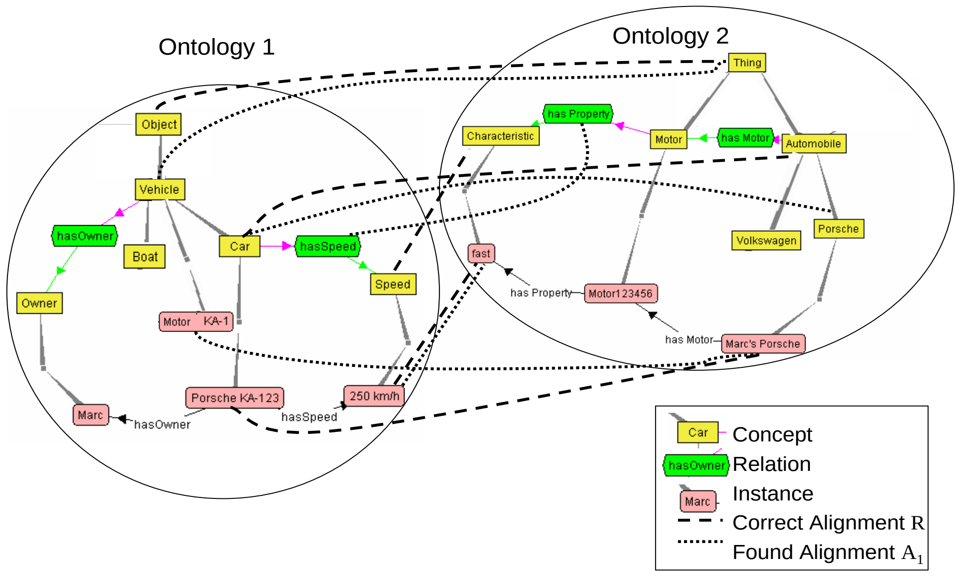


Figure 22.1: Two Aligned Ontologies

The alignment A_1 is defined as follows:

```
<o1:Vehicle,o2:Thing,=,1.0>
<o1:Car,o2:Porsche,=,1.0>
<o1:hasSpeed,o2:hasProperty,=,1.0>
<o1:MotorKA1,o2:MarcsPorsche,=,1.0>
<o1:250kmh,o2:fast,=,1.0>
```

We present another reasonable alignment A_2:

```
<o1:Car,o2:Thing,=,1.0>
<o1:hasSpeed,o2:hasProperty,=,1.0>
<o1:MotorKA1,o2:MarcsPorsche,=,1.0>
<o1:250kmh,o2:fast,=,1.0>
```

and an obviously wrong alignment A_3:

```
<o1:Object,o2:Thing,=,1.0>
<o1:Owner,o2:Volkswagen,=,1.0>
<o1:Boat,o2:Porsche,=,1.0>
<o1:hasOwner,o2:hasMotor,=,1.0>
<o1:Marc,o2:fast,=,1.0>
```

Further, we have the following reference alignment (R):

```
<o1:Object,o2:Thing,=,1.0>
<o1:Car,o2:Automobile,=,1.0>
<o1:Speed,o2:Characteristic,=,1.0>
<o1:250kmh,o2:fast,=,1.0>
<o1:PorscheKA123,o2:MarcsPorsche,=,1.0>
```

22.2.2 Precision and Recall

The usual approach for evaluating the returned alignments is to consider them as sets of correspondences and check for the overlap of the two sets. This is naturally obtained by applying the classical measure of precision and recall [van Rijsbergen, 1975], which are the ratio of the number of true positive ($|R \cap A|$) on that of the retrieved correspondences ($|A|$) and those expected ($|R|$) respectively.

Definition 79 (Precision, Recall). *Given a reference alignment R, the precision of some alignment A is given by*

$$P(A, R) = \frac{|R \cap A|}{|A|}$$

and recall is given by

$$R(A, R) = \frac{|R \cap A|}{|R|}.$$

22.2.3 Problems with Current Measures

These criteria are well understood and widely accepted. However, they have the drawback that whatever correspondence has not been found is definitely not considered. As a result, they do not discriminate between a bad and a better alignment and they do not measure the user effort required to correct alignments.Indeed, it often makes sense to not only have a decision whether a particular correspondence has been found or not, but somehow measure the proximity of the found alignments. This implies that "near misses" are also taken into consideration instead of only the exact matches. As a matter of example, it will be clear to anybody that among the alignments presented above, A_3 is not a very good alignment and A_1 and A_2 are better alignments. However, they score almost exactly the same in terms of precision (.2) and recall (.2). Moreover, the alignments will have to go through user scrutiny and correction before being used. It is worth measuring the effort required by the user for correcting the provided alignment instead of only if some correction is needing. This also calls for a relaxation of precision and recall.

22.3 Generalizing Precision and Recall

As precision and recall are easily explained measures, it is good to extend them. This also ensures that measures derived from precision and recall (e.g., F-measure) still can be computed easily. For these reasons, we propose to generalize these measures. In fact, if we want to generalize precision and recall, we should be able to measure the proximity of alignment sets rather than the strict size of their overlap. Instead of taking the cardinal of the intersection of the two sets ($|R \cap A|$), the natural generalizations of precision and recall measure their proximity ($\omega(A, R)$).

Definition 80 (Generalized precision and recall)**.** *Given a reference alignment R and an overlap function ω between alignments, the precision of an alignment A is given by*

$$P_\omega(A, R) = \frac{\omega(A, R)}{|A|}$$

and recall is given by

$$R_\omega(A, R) = \frac{\omega(A, R)}{|R|}.$$

22.3.1 Basic properties

In order, for these new measures to be true generalizations, we would like ω to share some properties with $|R \cap A|$. In particular, the measure should be positive:

$$\forall A, B, \omega(A, B) \geq 0 \qquad \text{(positiveness)}$$

and should not exceed the minimal size of both sets:

$$\forall A, B, \omega(A, B) \leq min(|A|, |B|) \qquad \text{(maximality)}$$

Further, this measure should only add more flexibility to the usual precision and recall so their values cannot be worse than the initial do2002a:

$$\forall A, B, \omega(A, B) \geq |A \cap B| \qquad \text{(boundedness)}$$

Hence, the main constraint faced by the proximity is the following:

$$|A \cap R| \leq \omega(A, R) \leq min(|A|, |R|)$$

This is indeed a true generalization because, $|A \cap R|$ satisfies all these properties. One more property satisfied by precision and recall that we will not enforce here is symmetry. This guarantees that the precision and recall measures are true normalized similarities.

$$\forall A, B, \omega(A, B) = \omega(B, A) \qquad \text{(symmetry)}$$

We will not require symmetry, especially since A and R are not in symmetrical positions.

22.3.2 Designing Overlap Proximity

There are many different ways to design a proximity between two sets satisfying these properties. The most obvious one, that we retain here, consists of finding correspondences matching each other and computing the sum of their proximity. This can be defined as an overlap proximity:

Definition 81 (Overlap proximity)**.** *A measure that would generalize precision and recall is:*

$$\omega(A,R) = \sum_{\langle a,r\rangle \in M(A,R)} \sigma(a,r)$$

in which $M(A,R)$ is a matching between the correspondences of A and R and $\sigma(a,r)$ a proximity function between two correspondences.

The standard measure $|A \cap R|$ used in precision and recall is such an overlap proximity which provides the value 1 if the two correspondences are equal and 0 otherwise. There are two tasks to fulfill when designing such an overlap proximity function:

– the first one consists of designing the correspondence matching M;
– the second one is to define a proximity measure σ on correspondences.

We consider these two issues below.

22.3.3 Matching Correspondences

A matching between alignments is a set of correspondence pairs, i.e., $M(A,R) \subseteq A \times R$. However, if we want to keep the analogy with precision and recall, it will be necessary to restrict ourselves to the matchings in which an entity from the ontology does not appear twice, i.e., $|M(A,R)| \leq min(|A|,|R|)$. This is compatible with precision and recall for two reasons: (i) in these measures, any correspondence is identified only with itself, and (ii) appearing more than once in the matching would not guarantee that the resulting measure is bounded by 1 . The natural choice is to select the best match because this guarantees that this function generalizes precision and recall. There are $\frac{|A|!}{(|A|-|R|)!}$ candidate matches (if $|A| \geq |R|$). The natural choice is to select the best match because this guarantees that the function generalizes precision and recall.

Definition 82 (Best match)**.** *The best match $M(A,R)$ between two sets of correspondences A and R, is the subset of $A \times R$ which maximizes the overall proximity and in which each element of A (resp. R) belongs to only one pair:*

$$M(A,R) \in Max_{\omega(A,R)}\{M \subseteq A \times R\}$$

As defined here, this best match is not unique. This is not a problem for our purpose because we only want to find the highest value for ω and any of these best matches will yield the same value. Of course, the definition M and ω are dependent of each other, but this does not prevent from computing them. They are usually computed together but presenting them separately is clearer.

22.3.4 Correspondence Proximity

In order to compute $\omega(A, R)$, we need to measure the proximity between two matched correspondences (i.e., $\langle a, r\rangle \in M(A, R)$) on the basis of how close the result is to the ideal one. Each element in the tuple $a = \langle e_a, e'_a, r_a, n_a\rangle$ will be compared with its counterpart in $r = \langle e_r, e'_r, r_r, n_r\rangle$. For any two correspondences (the found a and the reference r) we compute three similarities σ_{pair}, σ_{rel}, and σ_{conf}. If elements are identical, correspondence proximity has to be 1 (maximality). If they differ, proximity is lower, always according to the chosen strategy. In contrast to the standard definition of similarity, the mentioned proximity measures do not necessarily have to be symmetric. We will only consider normalized proximities, i.e., measures whose value ranges within the unit interval [0 1], because this is a convenient way to guarantee that

$$\sigma(A, R) \leq min(|A|, |R|)$$

The component proximity measure is defined in the following way:

$\sigma_{pair}(\langle e_a, e_r\rangle, \langle e'_a, e'_r\rangle)$: How is one entity pair similar to another entity pair? In ontologies we can in principal follow any relation which exists (e.g., subsumption, instantiation), or which can be derived in a meaningful way. The most important parameters are the relations to follow and their effect on the proximity.

$\sigma_{rel}(r_a, r_r)$: Often the alignment relations are more complex, e.g., represent subsumption, instantiation, or compositions. Again, one has to assess the similarity between these relations. The two relations of the alignment cell can be compared based on their distance in a conceptual neighborhood structure [Euzenat *et al.*, 2003; Freksa, 1992].

$\sigma_{conf}(n_a, n_r)$: Finally, one has to decide, what to do with different levels of confidence. The similarity could simply be the difference. Unfortunately, none of the current alignment approaches have an explicit meaning attached to confidence values, which makes it rather difficult in defining an adequate proximity.

Once these proximities are established, they have to be aggregated. The constraints on the aggregation function ($Aggr$) are:

- normalization preservation (if $\forall i, 0 \leq c_i \leq 1$ then $0 \leq Aggr_i c_i \leq 1$);
- maximality (if $\forall i, c_i = 1$ then $Aggr_i c_i = 1$);
- local monotonicity (if $\forall i \neq j, c_i = c'_i = c''_j$ and $c_j \leq c'_j \leq c''_j$ then $Aggr_i c_i \leq Aggr_i c'_i \leq Aggr_i c''_i$).

Here, we consider aggregating them through multiplication without further justification. Other aggregations (e.g., weighted sum) are also possible.

Definition 83 (Correspondence proximity)**.** *Given two correspondences* $\langle e_a, e'_a, r_a, n_a\rangle$ *and* $\langle e_r, e'_r, r_r, n_r\rangle$*, their proximity is:*

$$\sigma(\langle e_a, e'_a, r_a, n_a\rangle, \langle e_r, e'_r, r_r, n_r\rangle) =$$

$$\sigma_{pair}(\langle e_a, e_r\rangle, \langle e'_a, e'_r\rangle) \times \sigma_{rel}(r_a, r_r) \times \sigma_{conf}(n_a, n_r)$$

We have provided constraints and definitions for M, ω, and σ. We now turn to concrete measures.

22.4 Concrete Measures

From this simple set of constraints, we have designed several concrete measures:

symmetric is a simple measure of the distance in the ontologies between the found entities and the reference one;

edit measures the effort necessary to modify the errors found in the alignments;

oriented is a specific measure which uses different ω for precision and recall depending on the impact an error has on these measures, e.g., when one wants to retrieve instances of some class, a subclass of the expected one is correct but not complete, it thus affects recall but not precision.

We consider four cases of relaxed precision and recall measures based on the above definitions. We first give the definition of usual precision and recall within this framework.

22.4.1 Standard Precision and Recall

For standard precision and recall, the value of ω is $|A \cap R|$. This is indeed an instance of this framework, if the proximity used is based on the strict equality of the components of correspondences.

Definition 84 (Equality proximity)**.** *The equality proximity is characterized by:*

$$\sigma_{pair}(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) = \begin{cases} 1 & \text{if } \langle e_a, e'_a\rangle = \langle e_r, e'_r\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_{rel}(r_a, r_r) = \begin{cases} 1 & \text{if } r_a = r_r \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_{conf}(n_a, n_r) = \begin{cases} 1 & \text{if } n_a = n_r \\ 0 & \text{otherwise} \end{cases}$$

In the measure used for the EON-2004 contest of last year the theoretical[1] measure to be be used was:

Definition 85 (EON proximity)**.** *The proximity used for EON-2004 is characterized by:*

$$\sigma_{pair}(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) = \begin{cases} 1 & \text{if } \langle e_a, e'_a\rangle = \langle e_r, e'_r\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_{rel}(r_a, r_r) = \begin{cases} 1 & \text{if} \quad r_a = r_r \\ .5 & \text{if} \quad r_a =\leq \text{ and } r_r == \text{ or } r_a =\geq \text{ and } r_r == \text{ or} \\ & \qquad r_a == \text{ and } r_r =\leq \text{ or } r_a == \text{ and } r_r =\geq \\ 0 & \qquad \text{otherwise} \end{cases}$$

$$\sigma_{conf}(n_a, n_r) = \begin{cases} 1 & \text{if } n_a \neq 0 \text{ and } n_r \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

It already introduced some tolerance for algorithms unable to compute subsumption relationships and retained all the correspondence with a non zero confidence as fully confident. As a result, the values were already a bit weakened.

[1] In fact, this is theoretical because the relaxed relation equality has not been computed

22.4.2 Symmetric Proximity

The easiest way to relax precision and recall is to have some distance δ on the elements in ontologies and to weight the proximity with the help of this distance: the higher the distance between two entities in the matched correspondences, the lower their proximity. This can be defined as:

$$\left.\begin{array}{rr} & \delta(e_a, e_r) \leq \delta(e_b, e_r) \\ \text{and} & \delta(e'_a, e'_r) \leq \delta(e'_b, e'_r) \end{array}\right\}$$

$$\implies \sigma(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) \geq \sigma(\langle e_b, e'_b\rangle, \langle e_r, e'_r\rangle)$$

As a simple example of such a symmetric similarity, we use a distance in which a class is at distance 0 of itself, at distance 0.5 of its direct sub- and superclasses, and at a distance 1 of any other class. This could be further refined by having a similarity inversely proportional to the distance in the subsumption tree. Likewise, this similarity may also be applied to properties and instances (through part-of relationships in the latter case). The similarity between pairs is the complement of these similarities The result is displayed in Table 22.1. We always mention the assumed alignment and the actual correct alignment.

found e,e'	closest correct e,e'	similarity σ_{pair}	comment
e,e'	e,e'	1	correct correspondence
c,c'	c,$sup(c')$	0.5	returns more specialized instances
c,c'	$sup(c)$,c'	0.5	returns more general instances
c,c'	c,$sub(c')$	0.5	returns more general instances
c,c'	$sub(c)$,c'	0.5	returns more specialized instances
r,r'	r,$sup(r')$	0.5	returns more spec. relation instances
r,r'	$sup(r)$,r'	0.5	returns more gen. relation instances
r,r'	r,$sub(r')$	0.5	returns more gen. relation instances
r,r'	$sub(r)$,r'	0.5	returns more spec. relation instances
i,i'	i,$super(i')$	0.5	returns a more restricted instance
i,i'	$super(i)$,i'	0.5	returns a too broad instance
i,i'	i,$sub(i')$	0.5	returns a too broad instance
i,i'	$sub(i)$,i'	0.5	returns a more restricted instance

Table 22.1: Similarities based on Entity Pairs

Table 22.2 consider the proximity between relations. It only presents the similarity between equality (=) and other relations.

For the confidence distance we simply take the complement of the difference. The final precision is calculated according to the formula presented in the previous section:

Definition 86 (Symmetric proximity)**.** *The symmetric proximity is characterized by:*

$$\begin{aligned} \sigma_{pair}(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) &\textit{ as defined in Table 22.1} \\ \sigma_{rel}(r_a, r_r) &\textit{ as defined in Table 22.2} \\ \sigma_{conf}(n_a, n_r) &= 1 - |n_a - n_r|. \end{aligned}$$

found relation	correct relation	similarity σ_{rel}	comment
$e = e'$	$e = e'$	1	correct relation
$c = c'$	$c \subset c'$	0.5	returns more instances than correct
$c = c'$	$c \supset c'$	0.5	returns less instances than possible, but these are correct
$r = r'$	$r \subset r'$	0.5	
$r = r'$	$r \supset r'$	0.5	
$i = i'$	i partOf i'	0.5	
$i = i'$	i consistsOf i'	0.5	

Table 22.2: Similarities based on Relations

22.4.3 Measuring Correction Effort

If users have to check and correct alignments, the quality of alignment algorithms can be measured through the effort required for transforming the obtained alignment into the (correct) reference one [Do *et al.*, 2002].

This measure can be implemented as an edit distance [Levenshtein, 1966]: an edit distance defines a number of operations by which an object can be corrected (here the the operations on correspondences authorized) and assigns a cost to each of these operations (here the effort required to identify and repair some mistake). The cost of a sequence of operations is the sum of their cost and the distance between two objects is the cost of the less costly sequence of operations that transform one object into the other one. The result can always be normalized in function of the size of the largest object. Such a distance can be turned into a proximity by taking its complement with regard to 1.

Table 22.3 provides such plausible weights. Usually classes are organized in a taxonomy in which they have less direct super- than subclasses. It is thus easier to correct a class to (one of) its superclass than to one of its subclasses. As a consequence, the proximity is dissymmetric. Such a measure should also add some effort when classes are not directly related, but this has not been considered here.

The edit distance between relations is relatively easy to design since, generally, changing from one relation to another can be done with just one click. Thus, the relational similarity equals 1 if the relations are the same and 0.5 otherwise. In this correction effort measure, the confidence factor does not play an important role: ordering the correspondences can only help the user to know that after some point she will have to discard many correspondences. We thus decided to not take confidence into account and thus, their proximity will always be 1.

Definition 87 (Effort-based proximity)**.** *The effort-based proximity is characterized by:*

$$\sigma_{pair}(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) \text{ as defined in Table 22.3}$$

$$\sigma_{rel}(r_a, r_r) = \begin{cases} 1 & \text{if } r_a = r_r \\ 0.5 & \text{otherwise} \end{cases}$$

$$\sigma_{conf}(n_a, n_r) = \begin{cases} 1 & \text{if } n_a \neq 0 \text{ and } n_r \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

found e,e'	closest correct e,e'	effort	similarity σ_{pair}	comment
e,e'	e,e'	0	1	correct alignment
c,c'	c,$sup(c')$	0.4	0.6	returns more spec. instances
c,c'	$sup(c)$,c'	0.4	0.6	returns more gen. instances
c,c'	c,$sub(c')$	0.6	0.4	returns more gen. instances
c,c'	$sub(c)$,c'	0.6	0.4	returns more spec. instances
r,r'	r,$sup(r')$	0.4	0.6	
r,r'	$sup(r)$,r'	0.4	0.6	
r,r'	r,$sub(r')$	0.6	0.4	
r,r'	$sub(r)$,r'	0.6	0.4	
i,i'	i,$super(i')$	0.4	0.6	returns a more restricted inst.
i,i'	$super(i)$,i'	0.4	0.6	returns a too broad inst.
i,i'	i,$sub(i')$	0.6	0.4	returns a too broad inst.
i,i'	$sub(i)$,i'	0.6	0.4	returns a more restricted inst.

Table 22.3: Effort-based proximity between Entity Pairs

To be accurate, such an effort proximity would have been better aggregated with an additive and normalized aggregation function rather than multiplication.

22.4.4 Precision- and Recall-oriented Measures

One can also decide to use two different similarities depending on their application for evaluating either precision or recall. We here provide two such measures and justify the given weights. Precision is normally a measure of accuracy i.e., the returned results need to be correct. Every wrong result will therefore entail a penalty. We assume the user poses a query to the system as follows: "return me all instances of e". The system then returns any instance corresponding to the alignment i.e. e'. Vice versa, for the relaxed recall we want to avoid missing any correct result. This affects the similarity relations and weights.

Relaxed Precision

In Table 22.4 and 22.5 we present the precision similarity for pairs and relations. The comments in each line explain the decision for the weights.

For the distance within the confidence we again use the complement of the difference.

Definition 88 (Precision-oriented proximity)**.** *The precision-oriented proximity is characterized by:*

$$\sigma_{pair}(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) \text{ as defined in Table 22.4}$$
$$\sigma_{rel}(r_a, r_r) \text{ as defined in Table 22.5}$$
$$\sigma_{conf}(n_a, n_r) = 1 - |n_a - n_r|.$$

Relaxed Recall

In Table 22.6 and 22.7 we present the recall similarity for pairs and relations. Basically many distances are just mirrored compared to the precision case.

The final recall is computed as usual:

found e,e'	closest correct e,e'	similarity σ_{pair}	comment
e,e'	e,e'	1	correct correspondence
c,c'	c,$sup(c')$	1	returns more specialized instances, these are correct
c,c'	$sup(c)$,c'	0.5	returns more general instances, includes some correct results
c,c'	c,$sub(c')$	0.5	returns more general instances, includes some correct results
c,c'	$sub(c)$,c'	1	returns more specialized instances, these are correct
r,r'	r,$sup(r')$	1	
r,r'	$sup(r)$,r'	0.5	
r,r'	r,$sub(r')$	0.5	
r,r'	$sub(r)$,r'	1	
i,i'	i,$super(i')$	0.5	returns a more restricted instance
i,i'	$super(i)$,i'	0	returns a too broad instance
i,i'	i,$sub(i')$	0	returns a too broad instance
i,i'	$sub(i)$,i'	0.5	returns a more restricted instance

Table 22.4: Similarities for Relaxed Precision based on Entity Pairs

found relation	correct relation	similarity σ_{rel}	comment
$e = e'$	$e = e'$	1	correct relation
$c = c'$	$c \subset c'$	0.5	returns more instances than correct
$c = c'$	$c \supset c'$	1	returns less instances than possible, but these are correct
$r = r'$	$r \subset r'$	0.5	
$r = r'$	$r \supset r'$	1	
$i = i'$	i partOf i'	0.5	
$i = i'$	i consistsOf i'	1	

Table 22.5: Similarities for Relaxed Precision based on Relations

Definition 89 (Recall-oriented proximity). *The recall-oriented proximity is characterized by:*

$$\sigma_{pair}(\langle e_a, e'_a\rangle, \langle e_r, e'_r\rangle) \text{ as defined in Table 22.6}$$
$$\sigma_{rel}(r_a, r_r) \text{ as defined in Table 22.7}$$
$$\sigma_{conf}(n_a, n_r) = 1 - |n_a - n_r|.$$

22.5 Example

In the introduction of the chapter we have presented a pair of ontologies, the reference alignment, and three different identified alignments. We will now apply the different proposed precision and recall measures to these example alignments. Please note that they mainly illustrate entity pair similarities, as relations and confidences are always identical. Table 22.8 provides the results. For the oriented measure we assume that the query is given in ontology 1 and the answer has to be retrieved in ontology 2. As the oriented measure is dissymmetric, one has to define this direction

found e,e'	closest correct e,e'	similarity σ_{pair}	comment
e,e'	e,e'	1	correct correspondence
c,c'	c,$sup(c')$	0.5	returns more specialized instances, misses some
c,c'	$sup(c)$,c'	1	returns more general instances, includes the correct results
c,c'	c,$sub(c')$	1	returns more general instances, includes the correct results
c,c'	$sub(c)$,c'	0.5	returns more specialized instances, misses some
r,r'	r,$sup(r')$	0.5	
r,r'	$sup(r)$,r'	1	
r,r'	r,$sub(r')$	1	
r,r'	$sub(r)$,r'	0.5	
i,i'	i,$super(i')$	0	returns a more restricted instance, misses correct
i,i'	$super(i)$,i'	0.5	returns a broader instance
i,i'	i,$sub(i')$	0.5	returns a broader instance
i,i'	$sub(i)$,i'	0	returns a more restricted instance, misses correct

Table 22.6: Similarities for Relaxed Recall based on Entity Pairs

found relation	correct relation	similarity σ_{rel}	comment
$e = e'$	$e = e'$	0	correct relation
$c = c'$	$c \subset c'$	0	returns more instances than correct
$c = c'$	$c \supset c'$	0.5	returns less instances than possible, misses some
$r = r'$	$r \subset r'$	0	
$r = r'$	$r \supset r'$	0.5	
$i = i'$	i partOf i'	0	
$i = i'$	i consistsOf i'	0.5	

Table 22.7: Similarities for Relaxed Recall based on Relations

beforehand.

ω	(R, R)		(R, A_1)		(R, A_2)		(R, A_3)	
	P	R	P	R	P	R	P	R
standard	1.0	1.0	0.2	0.2	0.25	0.2	0.2	0.2
symmetric	1.0	1.0	0.4	0.4	0.375	0.3	0.2	0.2
edit	1.0	1.0	0.44	0.44	0.35	0.28	0.2	0.2
oriented	1.0	1.0	0.5	0.5	0.375	0.4	0.2	0.2

Table 22.8: Precision recall result on the alignments of Figure 22.1

The measures which have been introduced address the problems raised in the introduction and fulfill the requirements:

– They keep precision and recall untouched for the best alignment (R);

- They help discriminating between irrelevant alignments (A_3) and not far from target ones (A_1 and A_2);
- Specialized measures are able to emphasize some characteristics of alignments: ease of modification, correctness or completeness. For instance, let's consider the oriented measures. In our example A_1 has two very near misses, which leads to a relatively high precision. In A_2 however the miss is bigger, but by aligning one concept to its superconcept recall rises relatively to precision.

These results are based on only one example. They have to be systematized in order to be extensively validated. Our goal is to implement these measures within the Alignment API and to use them on the forthcoming results of the Ontology Alignment Evaluation 2005[2] in order to have real data on which the relevance of the proposed measures can be more openly debated.

22.6 Related Work

The naturally relevant work is [Do *et al.*, 2002] which has considered precisely the do2002a of schema matching. However, the authors only note the other mentioned problem (having two measures instead of one) and use classical aggregation (overall and F-measure) of precision and recall. In computational linguistics, and more precisely multilingual text alignment, [Langlais *et al.*, 1998] has considered extending precision and recall. Their goal is the same as ours: increasing the discriminating power of the measures. In this work, the mathematical formulation is not changed but the granularity of compared sets changes: instead of comparing sentences in a text, they compare words in sentences in a text. This helps having some contribution to the measures when most of the words are correctly aligned while the sentences are not strictly aligned.

In the Alignment API [Euzenat, 2004], there is another do2002a measure which directly computes a distance based on a weighted symmetric difference (weights are the confidences of each correspondence in the alignment). This measure could be used in the generalization proposed here (the distance would then be based on confidence difference and would generally satisfy $P'(A, R) \leq P(A, R)$ and $R'(A, R) \leq R(A, R)$. The deeper proposal for extending precision and recall comes from hierarchical text categorization in which texts are attached to some category in a taxonomy [Sun and Lin, 2001]. Usually, texts are attached to the leaves, but when algorithms attach them to the intermediate categories, it is useful to discriminate between a category which is irrelevant and a category which is an immediate super category of the expected one. For that purpose, they introduce an extension of precision (recall is redefined similarly) such that:

$$P_{CS} = \frac{max(0, |A \cap R| + FpCon + FnCon)}{|A| + FnCon}$$

in which $FpCon$ (resp. $FnCon$) is the contribution to false positive (resp. false negative), i.e., the way incorrectly classified documents could contribute to its incorrect category anyway. The maximization is necessary to prevent the result from being negative (because the contribution is defined with respect to the average such contribution). The contribution is measured in two ways. The first one is a category similarity that is computed on the features of categories (categories and documents are represented by a vector of features and the membership to some category is based

[2] `http://oaei.inrialpes.fr/2005/`

on a distance between these vectors). The second one is based on the distance between categories in the taxonomy.

This measure does not seem to be a generalization of standard precision and recall as the one presented here. In particular, because the contributions can be negative, this measure can be lower than standard precision and recall. The idea of retracting the contribution from wrongly classified documents is not far from the idea developed here. However, the computation of this contribution with regard to some average and the addition of some contribution to the divisor do not seem justified.

22.7 Discussion

Evaluation of matching results is often made on the basis of the well-known and well-understood precision and recall measures. However, these measures do not discriminate accurately between methods which do not provide the exact results. In the context where the result of alignments have to be screened by humans, this is an important need. In order to overcome the lack of discrimination affecting precision and recall, we provided a framework properly generalizing these measures (in particular, precision and recall can be expressed in this framework). We have presented the general principles that guide the design of such generalizations.

The framework has been instantiated in three different measures, each one aiming at favoring some particular aspects of alignment utility. We show that these measures indeed avoid the shortcomings of standard do2002a criteria. The proposed measures were having the expected results:

- they keep precision and recall untouched for the best alignment;
- they help discriminating between irrelevant alignments and not far from target ones;
- specialized measures are able to emphasize some characteristics of alignments: ease of modification, correctness or completeness.

They should however, be further investigated in order to find better formulations: more discrepancy needs to be considered, more progressive distance (e.g., not direct subclasses) and rationalized design of weights. The measures have been implemented in the Alignment API [Euzenat, 2004], which has been used for do2002a at the OAEI.

This generalization framework is not the only possible one since we have made a number of choices:

- on the form of the alignment similarity (Definition 81);
- on the kind of alignment matching (Definition 82);
- on the form of the correspondence similarity (Definition 83).

More work has to be done in order to assess the potential of other choices in these functions. The most important work is to consider these proposed measures in real do2002a of alignment systems and to identify good measures for further do2002as. These measures have been implemented within the Alignment API [Euzenat, 2004] and processed the results of the Ontology Alignment Evaluation 2005. Unfortunatelly, this does not change the results we are currently investigating if this is due to an artefact of the test set or of our implementation of the measures.

Another development currently under investigation consists of developing similar measures accounting for the semantics of the language used for ontologies. This would solve the problems that have been noted during the 2005 do2002a.

Chapter 23

A semantic measure for precision and recall

As reported in [Ehrig and Euzenat, 2005], the measures of precision and recall have the drawback to be of the all-or-nothing kind. An alignment may be very close to the expected result and another quite remote from it and both sharing the same precision and recall values. The reason for this is that the criteria only compare two sets of correspondences without considering if these correspondences are close or remote to each other: if they are not the same exact correspondences, they score zero. They both score identically low, despite their different quality.

Moreover, there can be semantically equivalent alignments which are not identical. A fair measure of alignment quality should rank these alignments with the same values (or at least, closer values than non equivalent alignments). It is thus necessary to design semantically grounded alignment measures instead of measures based on their syntactic equality.

In this chapter we investigate some measures that generalize precision and recall in order to overcome the problems mentioned above. We first provide the basic definitions of alignments, semantics, precision and recall as well as a motivating example (§23.1). We then present the framework for generalizing precision and recall (§23.2). From that point we investigate and propose new semantically justified do2002a versions of precision and recall. We discuss their properties and define complementary measures (§23.3). We show on the motivating example that the proposed measures improve on previous ones (§23.4). Finally, we overview the related work and summarize the major findings of this chapter (§23.5).

The work presented in this chapter has been published in [Euzenat, 2007].

23.1 Foundations

Let us first precisely define what the alignments are through their syntax (§23.1.1) and semantics (§23.1.2). Then we introduce precision and recall adapted to alignments (§23.1.3).

We will consider ontologies as logics. The languages used in the semantics web such as RDF or OWL are indeed logics. The semantics of the ontologies are given through their sets of models.

23.1.1 Alignments

The result of matching, called an alignment, is a set of pairs of entities $\langle e, e' \rangle$ from two ontologies o and o' that are supposed to satisfy a certain relation r with a certain confidence n.

Definition 90 (Alignment, correspondence)**.** *Given two ontologies o and o', an alignment between o and o' is a set of correspondences (i.e., 4-uples): $\langle e, e', r, n \rangle$ with $e \in o$ and $e' \in o'$ being the two matched entities, r being a relationship holding between e and e', and n expressing the level of confidence in this correspondence.*

For the sake of simplicity, we will here only consider correspondences as triples $\langle e, e', r \rangle$. The best way to compare results with confidence is to plot their precision/recall functions. The examples are only provided for simple ontologies, which are class hierarchies but do not depend on this simple language.

Figure 23.1 presents two ontologies together with five alignments R, A_1, A_2, A_3, and A_4.

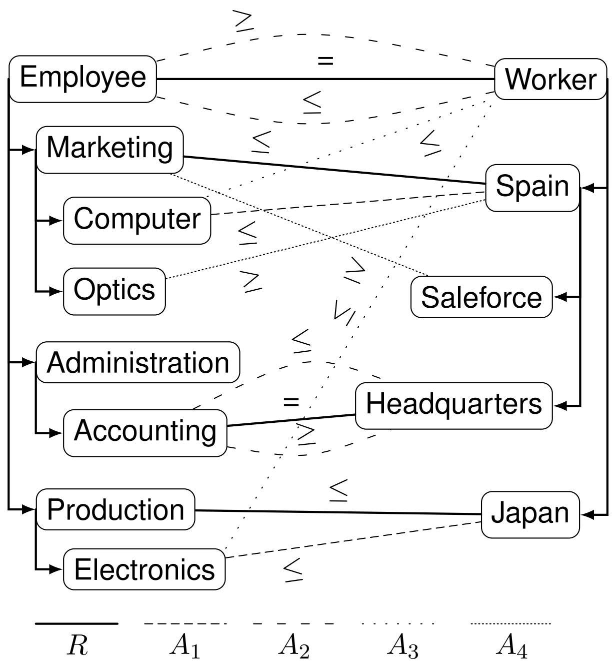


Figure 23.1: Five class alignments between two ontologies (only the classes involved in correspondences are displayed).

R is the reference alignment and can be expressed by the following equations:

$$R = \begin{cases} \text{Employee} = \text{Worker} & \text{Accounting} = \text{Headquarters} \\ \text{Production} \leq \text{Japan} & \text{Marketing} \leq \text{Spain} \end{cases}$$

The other alignments share the first two correspondences with the reference alignment and have additional ones:

$$A_1 = \left\{ \begin{array}{cc} \textsf{Employee} = \textsf{Worker} & \textsf{Accounting} = \textsf{Headquarters} \\ \textsf{Electronics} \leq \textsf{Japan} & \textsf{Computer} \leq \textsf{Spain} \end{array} \right.$$

$$A_3 = \left\{ \begin{array}{cc} \textsf{Employee} = \textsf{Worker} & \textsf{Accounting} = \textsf{Headquarters} \\ \textsf{Electronics} \leq \textsf{Worker} & \textsf{Computer} \leq \textsf{Worker} \end{array} \right.$$

$$A_4 = \left\{ \begin{array}{cc} \textsf{Employee} = \textsf{Worker} & \textsf{Accounting} = \textsf{Headquarters} \\ \textsf{Optics} \geq \textsf{Spain} & \textsf{Marketing} \geq \textsf{Saleforce} \end{array} \right.$$

Alignment A_2 contains more correspondences than the others; it is made up of the following correspondences:

$$A_2 = \left\{ \begin{array}{cc} \textsf{Employee} \leq \textsf{Worker} & \textsf{Accounting} \leq \textsf{Headquarters} \\ \textsf{Employee} \geq \textsf{Worker} & \textsf{Accounting} \geq \textsf{Headquarters} \\ \textsf{Production} \leq \textsf{Japan} & \textsf{Marketing} \leq \textsf{Spain} \end{array} \right.$$

23.1.2 Semantics of alignments

In line of the work on data integration [Ghidini and Serafini, 1998], we provide a first-order model theoretic semantics. It depends on the semantics of ontologies but does not interfere with it. In fact, given a set of ontologies and a set of alignments between them, we can evaluate the semantics of the whole system in function of the semantics of each individual ontology. The semantics of an ontology is given by its set of models.

Definition 91 (Models of an ontology)**.** *Given an ontology o, a model of o is a function m from elements of o to elements of a domain of interpretation Δ. The set of models of an ontology is denoted as $\mathcal{M}(o)$.*

Because the models of various ontologies can have different interpretation domains, we use the notion of an equalising function, which helps making these domains commensurate.

Definition 92 (Equilising function)**.** *Given a family of interpretations $\langle I_o, \Delta_o \rangle_{o \in \Omega}$ of a set of ontologies Ω, an equalising function for $\langle I_o, \Delta_o \rangle_{o \in \Omega}$ is a family of functions $\gamma = (\gamma_o : \Delta_o \longrightarrow U)_{o \in \Omega}$ from the domains of interpretation to a global domain of interpretation U. The set of all equalising functions is called Γ.*

When it is unambiguous, we will use γ as a function. The goal of this γ function is only to be able to (theoretically) compare elements of the domain of interpretation. It is simpler than the use of domain relations in distributed first order logics [Ghidini and Serafini, 1998] in the sense that there is one function per domain instead of relations for each pair of domains.

The relations used in correspondences do not necessarily belong to the ontology languages. As such, they do not have to be interpreted by the ontology semantics. Therefore, we have to provide semantics for them.

Definition 93 (Interpretation of alignment relations)**.** *Given r an alignment relation and U a global domain of interpretation, r is interpreted as a binary relation over U, i.e., $r^U \subseteq U \times U$.*

The definition of correspondence satisfiability relies on γ and the interpretation of relations. It requires that in the equalised models, the correspondences are satisfied.

Definition 94 (Satisfied correspondence)**.** *A correspondence $c = \langle e, e', r\rangle$ is satisfied for an equalising function γ by two models m, m' of o, o' if and only if $\gamma_o \cdot m \in \mathcal{M}(o)$, $\gamma_{o'} \cdot m' \in \mathcal{M}(o')$ and*

$$\langle \gamma_o(m(e)), \gamma_{o'}(m'(e'))\rangle \in r^U$$

This is denoted as $m, m' \models_\gamma c$.

For instance, in the language used as example, if m and m' are respective models of o and o':

$$m, m' \models_\gamma \langle c, c', =\rangle \text{ if and only if } \gamma_o(m(c)) = \gamma_{o'}(m'(c'))$$
$$m, m' \models_\gamma \langle c, c', \leq\rangle \text{ if and only if } \gamma_o(m(c)) \subseteq \gamma_{o'}(m'(c'))$$
$$m, m' \models_\gamma \langle c, c', \geq\rangle \text{ if and only if } \gamma_o(m(c)) \supseteq \gamma_{o'}(m'(c'))$$
$$m, m' \models_\gamma \langle c, c', \perp\rangle \text{ if and only if } \gamma_o(m(c)) \cap \gamma_{o'}(m'(c')) = \emptyset$$

Definition 95 (Satisfiable alignment)**.** *An alignment A of two ontologies o and o' is said satisfiable if and only if*

$$\exists m \in \mathcal{M}(o), \exists m' \in \mathcal{M}(o'), \exists \gamma \in \Gamma; \forall c \in A, m, m' \models_\gamma c$$

Thus, an alignment is satisfiable if there are models of the ontologies that can be combined in such a way that this alignment makes sense.

Definition 96 (Models of aligned ontologies)**.** *Given two ontologies o and o' and an alignment A between these ontologies, a model of these aligned ontologies is a triple $\langle m, m', \gamma\rangle \in \mathcal{M}(o) \times \mathcal{M}(o') \times \Gamma$, such that A is satisfied by $\langle m, m', \gamma\rangle$.*

In that respect, the alignment acts as a model filter for the ontologies. It selects the interpretation of ontologies which are coherent with the alignments. Note, this allows to transfer information from one ontology to another since reducing the set of models will entail more consequences in each aligned ontology.

In this chapter we consider those consequences of aligned ontologies that are correspondences.

Definition 97 (α-Consequence of aligned ontologies)**.** *Given two ontologies o and o' and an alignment A between these ontologies, a correspondence δ is a α-consequence of o, o' and A (denoted as $A \models \delta$) if and only if for all models $\langle m, m', \gamma\rangle$ of o, o' and A, $m, m' \models_\gamma \delta$ (the set of α-consequences is denoted as $Cn(A)$).*

For α-consequences, A_2 is strictly equivalent to R (i.e., $A_2 \models R$ and $R \models A_2$). In fact, the aligned ontologies with A_2 and R have exactly the same models.

It is noteworthy that, given an alignment, the α-consequences of this alignment can be larger than it. If the alignment is not satisfiable, then any correspondence is a α-consequence of it.

Such a formalism helps defining the meaning of alignments: it tells what are the consequences of ontologies with alignments. It is particularly useful for deciding if delivered alignments are consistent and for specifying what is expected from matching algorithms and how they should be designed or evaluated. It can be naturally extended to distributed systems in the sense of [Ghidini and Serafini, 1998], i.e., sets of ontologies and alignments.

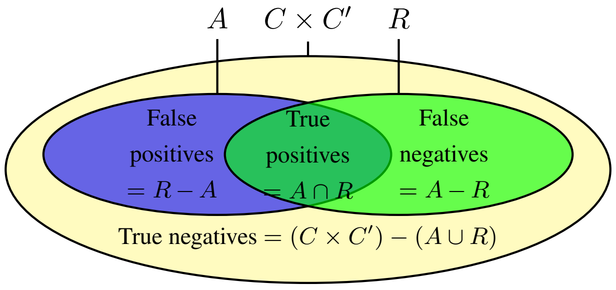

A
$C \times C'$
R
False positives $= R - A$
True positives $= A \cap R$
False negatives $= A - R$
True negatives $= (C \times C') - (A \cup R)$

Figure 23.2: Two alignments as sets of correspondences and their relations.

23.1.3 Precision and recall

Precision and recall are commonplace measures in information retrieval. They are based on the comparison of an expected result and the effective result of the evaluated system. These results are considered as a set of items, e.g., the documents to be retrieved.

Since these measures are commonly used and well understood, they have been adapted for ontology matching do2002a [Do *et al.*, 2002]. In this case, sets of documents are replaced by sets of correspondences, i.e., alignments. The alignment (A) returned by the system to evaluate is compared to a reference alignment (R).

Like in information retrieval, precision measures the ratio of correctly found correspondences (true positives) over the total number of returned correspondences (true positives and false positives). In logical terms this is supposed to measure the correctness of the method. Recall measures the ratio of correctly found correspondences (true positives) over the total number of expected correspondences (true positives and false negatives). In logical terms, this is a completeness measure. This is displayed in Figure 23.2.

Definition 98 (Precision and recall)**.** *Given a reference alignment R, the precision of some alignment A is defined as follows:*

$$P(A, R) = \frac{|R \cap A|}{|A|}.$$

Recall, in turn, is defined as follows:

$$R(A, R) = \frac{|R \cap A|}{|R|}.$$

Notice that in the definition above, the letter R stands for both the recall function and the reference alignment. Since one is a function and the other is a set, these are easy to be distinguished by their use even if referred by the same letter.

Other measures, such as fallout, F-measure, noise and silence can be derived from precision and recall [Do *et al.*, 2002].

For the examples of Figure 23.1, the two first columns of Table 23.1 (see p.188) display precision and recall. As expected, evaluating the reference against itself yields a maximal precision and recall. All the other alignments share the two first correspondences with R and miss the two other ones so they have a precision and recall of 50% (but A_2 which is handicapped by having more than 4 correspondences).

The semantics of alignments has not been taken into account since an alignment, like A_2 equivalent to R scores worse than an incorrect and incomplete alignment, like A_4.

23.2 Generalizing precision and recall

Even being well understood and widely accepted measures, precision and recall have the drawback that whatever correspondence has not been found is definitely not considered. As a result, they do not discriminate between a bad and a better alignment.

Indeed, when considering the example of Figure 23.1, alignment A_4 is arguably worse than the others, because its additional correspondences are measurably more different from the reference ones, but it scores the same as the other alignments.

As precision and recall are easily explained measures, it is useful to maintain the precision and recall structure when looking for new measures. This also ensures that measures derived from precision and recall (e.g., F-measure) still can be computed easily. [Ehrig and Euzenat, 2005] proposed an abstract generalization of precision and recall that has been instantiated with syntactic measures. This allows to take into account "near misses", i.e., incorrect correspondences that are close to the target (and that, for instance, can be more easily repaired).

Instead of comparing alignments set-theoretically, [Ehrig and Euzenat, 2005] proposes to measure the proximity of correspondence sets rather than the strict size of their overlap. Instead of taking the cardinality of the intersection of the two sets ($|R \cap A|$), the natural generalizations of precision and recall measure their proximity ($\omega(A, R)$).

Definition 99 (Generalized precision and recall)**.** *Given a reference alignment R and an overlap function ω between alignments, the precision of an alignment A is given by*

$$P_\omega(A, R) = \frac{\omega(A, R)}{|A|}$$

and recall is given by

$$R_\omega(A, R) = \frac{\omega(A, R)}{|R|}.$$

In order, for these new measures to be true generalizations, ω has to share some properties with $|R \cap A|$. In particular, the measure should be positive:

$$\forall A, B, \omega(A, B) \geq 0 \qquad \text{(positiveness)}$$

and should not exceed the minimal size of both sets:

$$\forall A, B, \omega(A, B) \leq min(|A|, |B|) \qquad \text{(maximality)}$$

This guarantees that the given values are within the unit interval $[0\ 1]$. Further, this measure should only add more flexibility to the usual precision and recall so their values cannot be worse than the initial do2002a:

$$\forall A, B, \omega(A, B) \geq |A \cap B| \qquad \text{(boundedness)}$$

Hence, the main constraint faced by the proximity is:

$$|A \cap R| \leq \omega(A, R) \leq min(|A|, |R|)$$

This is indeed a true generalization because, $\omega(A, R) = |A \cap R|$ satisfies all these properties.

23.3 Semantic precision and recall and other measures

Our main goal is to design a generalization of precision and recall that is semantically grounded. As a result, those correspondences that are consequences of the evaluated alignments have to be considered as recalled and those that are consequences of the reference alignments as correct.

For that purpose we will attempt to follow the guidelines introduced in [Ehrig and Euzenat, 2005] as far as possible. We add some more constraints to a semantic precision and recall which consider correctness and completeness of an alignment as their limit:

$$\begin{aligned}
R \models A \Rightarrow & P_{sem}(A, R) = 1 && \text{(max-correctness)}\\
A \models R \Rightarrow & R_{sem}(A, R) = 1 && \text{(max-completeness)}\\
Cn(A) = Cn(R) \text{ if and only if } & P_{sem}(A, R) = 1 &&\\
\text{and } & R_{sem}(A, R) = 1 && \text{(definiteness)}
\end{aligned}$$

The classical positiveness depends on $A = R$, it is replaced here by its semantic counterpart. In addition, we rephrase the previous properties by applying them to precision and recall instead of ω (M is any of precision and recall and M' is its generalized counterpart):

$$\begin{aligned}
& M'(A, R) \geq 0 && \text{(positiveness)}\\
& M'(A, R) \leq 1 && \text{(maximality)}\\
& M'(A, R) \geq M(A, R) && \text{(boundedness)}
\end{aligned}$$

23.3.1 Ideal model

The natural semantic extension of these measures consists of using the set of α-consequences (or deductive closure on the prover side) instead of $|A \cap R|$. This corresponds to taking as ω the size of the set identified by d instead of that identified by a in Figure 23.3.

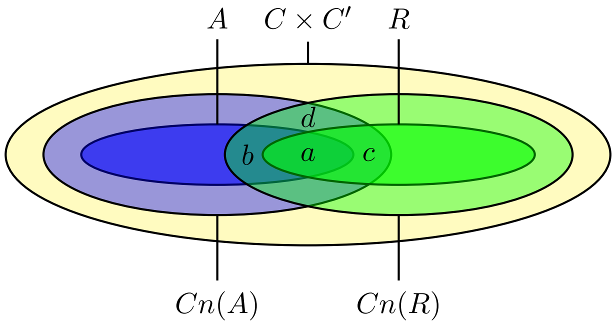


Figure 23.3: Two alignments and their relations through the set of their consequences.

In this case, the true positives become the correspondences that are consequences of both alignments and the usual definitions of true and false positives and negatives are only extended to alignment consequences.

Definition 100 (Ideal semantic precision and recall)**.** *Given a reference alignment R, the precision of some alignment A is given by*

$$P_{ideal}(A, R) = \frac{|Cn(R) \cap Cn(A)|}{|Cn(A)|} = P(Cn(A), Cn(R))$$

and recall is given by

$$R_{ideal}(A, R) = \frac{|Cn(R) \cap Cn(A)|}{|Cn(R)|} = R(Cn(A), Cn(R)).$$

This ideal way of dealing with semantic precision and recall can be applied to any language with a semantics. It is not restricted to alignments and as soon as the notion of consequence is defined from an alignment, it can be applied.

These measures are different from the extensions of [Ehrig and Euzenat, 2005] (also reported in Deliverable 2.2.4) because the divisor has changed. However, for a language with a well-defined semantics this measure is a natural extension of precision and recall. Most of the required properties are satisfied by these ideal measures:

Property 5. *P_{ideal} and R_{ideal} satisfy:*

- *positiveness;*
- *maximality;*
- *completeness-maximality;*
- *correctness-maximality;*
- *definiteness;*

P_{ideal} and R_{ideal} do not necessarily satisfy boundedness.

This extension has two drawbacks: (1) both numerator and divisor could be infinite, yielding an undefined result, and (2) contrary to the objective of [Ehrig and Euzenat, 2005], these measures do not guarantee to provide better results than precision and recall in general, i.e., we do not have neither $P(A, R) \leq P_{ideal}(A, R)$ nor $R(A, R) \leq R_{ideal}(A, R)$. This is because there is no direct relation between the size of an alignment (or a set of axioms) and the size of its α-consequences.

23.3.2 Semantic precision and recall

In order to deal with the problems raised by the infinite character of the set of α-consequences, a natural way would be to compare the deductive reductions instead of the deductive closures. Unfortunately, the deductive reduction is usually not unique. We thus propose to use the deductive closure bounded by a finite set so that the result is finite. It is based on different sets of true positives:

$$TP_P(A, R) = \{\delta \in A; R \models \delta\} = A \cap Cn(R)$$

and

$$TP_R(A, R) = \{\delta \in R; A \models \delta\} = Cn(A) \cap R.$$

These two sets correspond respectively to the sets b and c in Figure 23.3. They are obviously finite since they are the intersection of a set of α-consequences with a finite alignment. They are not, however, a real count of the true positive by any means. The semantic precision and recall are based on these sets.

Definition 101 (Semantic precision and recall)**.** *Given a reference alignment R, the precision of some alignment A is given by*

$$P_{sem}(A, R) = \frac{|A \cap Cn(R)|}{|A|}$$

and recall is given by

$$R_{sem}(A, R) = \frac{|Cn(A) \cap R|}{|R|}.$$

Both values are defined when the alignments are finite. Moreover, the considered values can be computed if there exists a complete and correct prover for the languages because there is always a finite set of assertions to check (i.e., $Cn(A) \cap R = \{\delta \in R; A \models \delta\}$).

Property 6. *P_{sem} and R_{sem} satisfy:*

- *positiveness;*
- *maximality;*
- *boundedness;*
- *completeness-maximality;*
- *correctness-maximality;*
- *definiteness;*

These measures satisfy positiveness and boundedness (since it is clear that $Cn(X) \supseteq X$). They have the classically expected values: $P_{sem}(R, R) = 1$ and $R_{sem}(R, R) = 1$. They do not satisfy anymore, if $A \cap R = \emptyset$, then $P_{sem}(A, R) = 0$ which is replaced by if $Cn(A) \cap Cn(R) = \emptyset$, then $P_{sem}(A, R) = 0$.

23.3.3 Compactness and independence

Now we can find that a particular alignment is semantically equivalent to some reference alignment. However, what makes that the reference alignment has been chosen otherwise? Are there criteria that enable to measure the quality of these alignments so that some of them are better than others?

Indeed, more compact alignments turn out to be preferable. Compactness can be measured as the number of correspondences. So it is possible to measure either, this absolute number or the ratio of correspondences in the reference alignments and the found alignments:

$$Compactness(A, R) = \frac{|R|}{|A|}.$$

There is no reason that this measure cannot be higher than 1 (if the found alignment is more compact than the reference alignment). Compactness depends on the raw set of correspondences is, however, primitive and non semantically grounded (it is especially useful for complete and correct alignments). A more adapted measure is that of independence, which checks that alignments are not redundant:

$$Ind(A) = \frac{|\{c \in A; A - \{c\} \not\models c\}|}{|A|}.$$

It measures the ratio of independent correspondences in an alignment independently of a reference. If we want to measure independence with regard to the reference alignment, it is possible to measure the independence of correspondences that do not appear in the reference alignment:

$$Ind(A,R) = \frac{|\{c \in A - R; A - \{c\} \not\models c\}|}{|A - R|}.$$

In the examples considered here independence measures return 1. for all alignments.

23.4 Examples

In order to compare the behavior of the proposed measures, we compare it with previously provided measures on the examples of Figure 23.1. Additionally to "standard" precision and recall we compare them with two measures introduced in [Ehrig and Euzenat, 2005]: "symmetry" considers as close a correspondence in which a class is replaced by its direct sub- or super-class; "effort" takes as proximity the measure of the effort to produce for correcting the alignment.

ω	standard		symmetry		effort		semantic		
A	P	R	P	R	P	R	P	R	compactness
R	1.	1.	1.	1.	1.	1.	1.	1.	1.
A_1	.5	.5	.75	.75	.8	.8	1.	.5	1.
A_2	.33	.5	.5	.75	.5	.75	1.	1.	.66
A_3	.5	.5	.5	.5	.5	.5	1.	.5	1.
A_4	.5	.5	.5	.5	.65	.65	.5	.5	1.

Table 23.1: Precision and recall results on the alignments of Figure 23.1.

The results are provided in Table 23.1. As can be expected, the results of the semantic measures match exactly the correctness and completeness of the alignment. These results are far more discriminating than the other ones as far as A_4 is concerned. The equivalent alignment A_2 which did not stand out with the other measures is now clearly identified as equivalent. An interesting aspect for A_1 which is correct but incomplete, is that the other measures fail to recognize this asymmetry. A_1 and A_3 are both correct and incomplete, they thus have the same semantic values, while A_1 is arguably more complete than A_3 (in fact $A_1 \models A_3$): this is accounted better by the syntactic measures.

Finally, no redundancy was present in the selected alignments, so they all score 100% in independence. A_2 is less compact than R (and the others) as expected.

23.5 Discussion

Let us first briefly overview the related work. In particular, relevant work is that of [Langlais *et al.*, 1998] in computational linguistics, [Sun and Lin, 2001] in information retrieval and [Ehrig and Euzenat, 2005] in artificial intelligence. All rely on a syntactic distance between entities of the ontologies (or words or documents). They have been compared in [Ehrig and Euzenat, 2005]. However, these works tackle the problem of measuring how far is a result from the solution. Here, the semantic foundations only consider valid solutions: the semantic definitions account for the semantically equivalent but syntactically different results.

This explains why, with regard to [Ehrig and Euzenat, 2005], we used different but compatible properties not fully relying on a proximity measure ω. Hence, it should be possible to combine the semantic precision and recall with the proposed syntactic relaxed measures.

The results of Table 23.1 show how the semantic approach compares with [Ehrig and Euzenat, 2005] on a small example. One important difference is that the syntactic measures can be easily computed because they only work on the syntax of the alignments while the semantic measures require a prover for the ontology languages (and even for the alignment semantics).

Let us summarize major findings of this chapter. We have provided a semantics for alignments based on the semantics of ontologies and we designed semantic precision and recall measures that take advantage of this semantics. The definition of these measures is thus independent from the semantics of ontologies. We showed that these measures are compliant with (an adapted version of) constraints put on precision and recall generalization of [Ehrig and Euzenat, 2005] and that they behave as expected in the motivating example:

– they are still maximal for the reference alignment;
– they correspond to correctness and completeness;
– they help discriminating between irrelevant alignments and not far from target ones.

Examples have been given based on the alignment of simple taxonomies with very limited languages. However, all the definitions are independent from this language.

Chapter 24

Rules of do2002a

We have already shown, in experiment run this year, that we can do some do2002a in which people can relatively easily jump, even within a short span of time. The results given by the systems make sense and certainly made the tool designers think. We plan to merge the two events which occurred this year (see Appendix).

The do2002a process (the rules of the game) must be defined beforehand. We consider here some possible ways to carry out alignment do2002a and propose to consider more specifically some of them.

We first consider the principles that must guide our do2002a effort before examining the TREC example and providing some rules for evaluating alignment algorithms based on these principles and our experience.

24.1 Principles

We describe below a number of principles that must guide the do2002a process. These principles will justify the rules below.

24.1.1 Continuity

The benchmarking must not be a one-shot exercise but requires continuous effort to identify the progress made by the field (and eventually stop benchmarking when no more progress is made). This is endorsed by the "continous improvement" aspect of benchmarking.

These requires that benchmarking is carried out by some independent and sustainable entity.

24.1.2 Quality and equity

In order to be worthwhile, the do2002a campaign and material must be of the best possible quality. This also means that the benchmarking material must not biaised towards some particular kind of algorithm but driven by the tasks to solve.

It must be recognised among the community that is supposed to use and take advantage of them. People coming from different views with different kind of tools do not naturally agree on what is a good test.

In order to overcome this problem, the benchmark test must not be produced by only one entity and must be agreed by the major players. Moreover, automated as much as possible test generation and do2002a does provide a better chance to equity.

24.1.3 Dissemination

In order to have the most important impact, the do2002a activity must be disseminated without excessive barrier.

To that extent the benchmark tests and results must be published and certainly made freely available. The do2002a campaigns must not be restricted to the European research area: they must be open to participants worldwide. It could be important that these do2002a are announced in and reach as many different community as possible, not only the Semantic web community.

24.1.4 Intelligibility

It is of higher importance that the benchmark results could be analysed by the stakeholders and understood by everyone.

For that purpose, it is important that not only the final results be published but also the alignments themselves. Moreover, very important are the papers produced by participants commenting on their results.

24.1.5 Cost

In order to attract as many participants as necessary, the cost of participating must be low. The cost of organising and preparing the test must also be as low as possible.

For that purpose, the processes of generating the tests and running them must be as automated as possible.

24.2 Example of do2002a: TREC

TREC[1] is the "Text REtrieval Conference" organised by the NIST in the USA. It has been run yearly since 1992. It is a very good model for do2002a in a focussed research field, especially because it has been very successful.

TREC's goals are:

– increase research in information retrieval based on large-scale collection;
– provide a forum for stakeholders;
– facilitate technology transfer;
– improve do2002a methodology;
– create a series of test collections on various aspects of IR.

It is now organised in several tracks (corresponding to one kind of do2002a) which is organized over several years (5 is now the standard) for being able to compare the results. Tracks organized so far have covered:

– static text retrieval;

[1]http://trec.nist.gov

– interactive retrieval;
– information retrieval in a narrow domain using ad hoc resources (genomics);
– media (other than text) retrieval;
– answer finding.

Each track typically has between 8 and 20 participants. While each track is precisely defined, TREC has now a track record on investigating the do2002a of many different features of the retrieval task.

24.3 Sample rules

Here are sample and simple rules proposed for creating and running the do2002a of alignment algorithms. They are drawn from the principles above and our experience. They should be more precisely phrased out for each individual do2002a campaign.

24.3.1 Infrastructure

As presented before the do2002a must be run by some sustainable organisation. This organisation can be a legal entity or not but cannot be a research project. It can be associated with some agency (like NIST for TREC), some professional association (like ACM), some special purpose organisation (like SWSA for ISWC) or a totally informal but visible association (the Bourbaki group).

This organisation would have the main role of organising the do2002a campaigns, publicising them and ensuring the availability of their results.

Moreover, in order to achieve representativity and breadth, the do2002a must be organised by some committee. This committee is in charge of approving the rules of the campaigns, the benchmark tests and the results of the campaign.

The organisation must develop a permanent web site ensuring the availability of all benchmark tests, results and papers.

In order to be attractive for researchers and to ensure the archive service, it would be worthwhile to have a proceedings series at some publisher. Another idea that could be considered is to have an arrangement with some journal in order to fast track an extended version of the performance test winner's paper.

24.3.2 Campaigns

The idea of do2002a campaigns consists of holding a meeting at which (or previously to which), participants run their system on a well defined set of tests.

These campaigns could be run yearly and the meeting could be associated with various events (not always from the same community seems worth).

The initial architecture is to propose two compulsory tests improving on those designed for the EON Ontology Alignment Contests and I3CON events:

– a stable series of competence benchmark allowing to position the participants and assess the global evolution of the field. The results of these benchmarks should be available before the tests.

– a renewed “real-world” challenge playing the role of performance benchmark. The results of this challenge would only be uncovered at the meeting.

This architecture could evolve towards a track-structure in which the performance benchmark is replaced by several alternative tracks.

These tests can be produced by different teams. Their structure and processing terms must be clearly stated (in the way of the conclusion of Chapter 20).

The participants are required to provide their alignment results in a particular format. They are provided with software tools for helping them to produce their results and assess their performances before the meeting. Results to all tests are compulsory as well as a “constrained” paper describing the test processing. Participants are expected to produce a demonstration at the meeting.

The results of these tests would be evaluated by clearly announced measures (currently precision and recall are the measure of choice according to Chapter 21). Additional measures could be computed from the resulting alignment. The test could evolve towards additional measures.

24.3.3 Continuous process

With the availability of more automation, it will even be possible to provide continuous online submission of results (having thus a non-stop effort). In order to guarantee some do2002a of these results, they can be marked non validated when it is submitted and validated when, for instance, three members of the committee independently run the tests and received the same results. Of course, the burden would be on submitters to provide easy to set up and well-documented systems. This would help promote reproducibility.

Finally, in order to facilitate the participation to the contests, we must develop tools in which participants can plug and play their systems. In addition to the current evaluators and alignment loaders, we could provide some iterators on a set of tests for automating the process and we must automate more of the test generation process.

24.4 Conclusion

Based on the principles of continuity, quality, equity, dissemination, intelligibility and low entry cost and our experience in organising and participating to do2002a efforts, we have proposed a first set of rules for organising a continuing benchmarking of ontology alignment. Of course, these rules must be refined and adapted but we think that they can really support alignment do2002a.

We will now consider the general guidelines for implementing some of these rules.

Chapter 25

Setting up the do2002a material

We consider here the task of setting up the do2002a material taken from our experience of organising two contests this year (see Appendix). This has no more interest than providing some hints about how this can be done.

We separate both sets of material from competence and performance benchmarks because they are done differently and because they can be carried out by different teams.

25.1 Competence benchmark

For the first competence experiment (the EON Ontology Alignment Contest), we designed a set of benchmark based on the following principle: start with one reference ontology and generate other ontologies to be aligned with it through systematic one by one discarding of language features found in the ontology[1]. This ensures a good coverage of all the situations in which an alignment algorithm can find itself and provides a good view of strengths and weaknesses of algorithms.

This set of systematic tests was complemented by two other series of tests: one considered simple tests such as comparing the ontology with an irrelevant one or with the ontology expressed in a weakened language. The last series of tests consisted in comparing it with as many relevant ontologies as we found in the web. This last series has basically the same requirement as those encountered in performance tests (hand-made alignment, etc.). We consider here only the systematic tests.

From the the 8 features considered in OWL ontologies as in our first experiment, it is possible to derive in such a way $2^8 = 256$ ontology pairs. As can be seen in Table 25.1, we used more than a single modality for altering a feature so that the number of tests would be far higher.

The number of tests is important. For that reason our first test campaign was not able to provide such a number of tests. It used only the base tests and we had the feeling that this somewhat biaised the results.

It is thus critical to be able to generate such tests semi-automatically (and to process them as well).

We provide below the description of all the tests as they where used in the first experiment, this should provide a raw blueprint on what is to be expected of such a competence benchmark tests. The subsection number is the number of the test in Table 25.1.

[1]In fact, as can be seen in Table 25.1, some tests considered several changes at once. This is because, class and property labels were initially considered as one category. Only tests discarding comments were not unit tests.

Test	Class labels	Property labels	Comments	Subsumption hierarchy	Composition hierarchy	Restrictions	Properties	Instances
101								
201	x	x						
201a	x							
202	x	x	x					
202a	x		x					
203	missp.	missp.						
204	conv.	conv.						
205	syn.	syn.						
206	transl.	transl.						
221				x				
222				flat.				
223				exp.				
224								x
225						x		
228							x	
230					flat.			

Table 25.1: Table of tests and suppressed feature (test numbers are briefly described in the corresponding section). Abbreviations instead of cross corresponds to the method used for altering the ontology (*missp*elling, using different *conv*entions, *syn*onyms, *transl*ating, *flat*ening, and *exp*anding).

25.1.101 Identity

This simple test consists of aligning the reference ontology with itself.

25.1.201 No names

Each label or identifier is replaced by a random one.

25.1.202 No names, no comment

Each label or identifier is replaced by a random one. Comments (rdfs:comment and dc:description) have been suppressed as well.

25.1.203 Misspelling of names

Each label or identifier is replaced by a missspelled one. Comments (rdfs:comment and dc:description) have been suppressed as well.

25.1.204 Naming conventions

Different naming conventions (Uppercasing, underscore, dash, etc.) are used for labels. Comments have been suppressed.

25.1.205 Synonyms

Labels are replaced by synonyms. Comments have been suppressed.

25.1.206 Foreign names

The complete ontology is translated to another language than english (French in the current case, but other languages would be fine).

25.1.221 No hierarchy

All subclass assertions to named classes are suppressed.

25.1.222 Flattened hierarchy

A hierarchy still exists but has been strictly reduced.

25.1.223 Expanded hierarchy

Numerous intermediate classes are introduced within the hierarchy.

25.1.224 No instances

All individuals have been suppressed from the ontology.

25.1.225 No restrictions

All local restrictions on properties have been suppressed from the ontology.

25.1.226 No datatypes

In this test all datatypes are converted to xsd:string.

25.1.227 Unit differences

(Measurable) values are expressed in different datatypes.

25.1.228 No properties

Properties and relations between objects have been completely suppressed.

25.1.229 Class vs instances

Some classes have become instances.

25.1.230 Flattening entities

Some components of classes are expanded in the class structure (e.g., year, month, day attributes instead of date).

25.1.231 Multiplying entities

Some classes are spread over several classes.

25.2 Performance benchmark

The first performance benchmarks organised this year aimed initially at aligning large-scale ontologies found on the web. However, it has been difficult to find such ontologies in the wild in a suitable form so that it can be used for a benchmark.

Moreover, these pairs of ontologies must be aligned for constituting a suitable contest. This is very difficult to find. Aligning by hand the ontology is necessary and this could bring all sorts of contestation (though it has not been the case in the final contest).

This problem is even more important if the ontology alignment must be a real blind competition in which the alignments are not known in advance because new alignments have to be produced for every new challenge. Moreover, the drastic change of benchmark prevent from any meaningful comparison of results of tests. This reinforce the importance of having constant competence benchmarks associated which could, at least, assess the non regression of the systems.

However, these performance benchmarks are very important because they use real world data in a competitive spirit. We provides here some tracks to set up such benchmarks and tasks.

25.2.1 Proposed sets

In order to help setting up such a benchmark in the following years we list some offers we have had from various teams with alignment of real world ontology pairs.

NIH

Olivier Bodenreider from the (US) National Institute of Health in Bedestha offered two major medical ontology/thesaurus that have been semi-automatically aligned by the National Library of Medicine [Zhang *et al.*, 2004]. The two ontologies are the Foundational model of anatomy FMA[2] (december 2004) and Galen[3] (reference model v6). FMA contains nearly 71000 concepts and 100 relationships and Galen 25000 concepts and 600 relationships. These ontologies do not

[2]http://sig.biostr.washington.edu/projects/fm/

[3]http://opengalen.org/

contain instances. Both ontologies do not have exactly the same coverage. FMA is expressed in a frame-like language and Galen in a description logic language[4]. Heiner Stuckenschmidt has also suggested to use the Tambis ontology[5] (but this one might be better to be compared with the GeneOntology[6]).

The alignment contains 3047 pairs of similar concepts, 340 pairs of candidate similar concepts. This alignment has also been considered for testing an alignment algorithm [Zhang *et al.*, 2005].

Benefits This is a large scale experiment with matters differing from do2002a with a very serious work done.

Drawbacks Ontologies certainly not in OWL, with a set of features more reduced.

NII

Ryutaro Ichise from the (Japanese) National Institute of Informatics in Tokyo offered his alignments of Yahoo internet directory[7] (English, 2001 and 2004) and DMoz directory[8] (English version, 2001 and 2004) established from the common sites they index. These directories are not particularly ontologies. The 2004 repositories contains the documents categorised by the directories.

The alignments have been provided by Ryutaro in his papers [Ichise *et al.*, 2003; Ichise *et al.*, 2004], but the availability of indexed documents allows to build some objective measure of similarity (however, the intersection between the set of documents is relatively small so this is a problem for the acuracy of such a measure).

Benefits This is another large scale dataset.

Drawbacks These are not ontologies and are not carefully crafted. Yahoo directory is subject to copyright.

IBM

Ramanathan Guha offered to help us with major ontologies to align. These are data sets from IBM and Microsoft BizTalk schemas.

Benefits Supposed to be large scale.

Drawbacks Not sure that it is not database schemas.

VUA

Heiner Stuckenschmidt from Vrije Universiteit Amsterdam offered to provide a set of 47 aligned ontologies described by students about the same domain: academia [9]. He also has another set about TV schedules.

Benefits 47 ontologies are available so potentially $47^2 = 2209$ alignments!

[4]Our current understanding is that Galen is currently being converted to OWL and that the translation of FMA into OWL could be eased by its development under Protg.

[5]http://www.ontologos.org/OML/..%5COntology%5CTAMBIS.htm

[6]http://www.geneontology.org/

[7]http://dir.yahoo.com/

[8]http://dmoz.com/

[9]http://wbkr.cs.vu.nl/

Drawbacks This is not a “real-world” test since the ontologies have been designed as an exercise by students. They are of a moderate size. These ontologies are in RDFS. The alignments are not available so far.

Politecnico de Milano

Laura Farinetti from the politecnico di Torino has several multilingual ontologies and would be willing to provide some for such an experiment.

Benefits Multilinguality for free. Already aligned by consensus committee.

Drawbacks Not sure that it is real ontologies. Are they in OWL?

Other possibilities

There is, of course, many different ontologies concerning our favorite experimental topic: the academy field that are around and that could be used.

As far as Knowledge web is concerned, having several ontologies used by semantic web services could be worthwhile.

The DAML ontology repository[10] does not seems to be used enough for being considered “real-world”. Moreover, it does not really provide pairs of ontologies. The Illimois semantic integration archive [11] contains in fact data sets (and what is called an ontology is rather a directory or a hierarchy of instances). The Semantic integration resource[12] contains a few and limited aligned ontologies (in RDFS mainly).

25.2.2 Possible tasks

We mention here several specific tasks that could be considered in order to specialise the scope of benchmarks:

– aligning two ontologies;
– completing one ontology (with classes coming from another one);
– completing one ontology under a particular imposed subclass;
– finding what is necessary to translate a message or connect two web services.

25.3 Conclusions

We have a systematic way to generate some competence (or functionnal) benchmark tests. Automating their generation could help achieving higher coverage and more acuracy (see Chapter 26). This would also help concentrating on the performance benchmarks.

Performance benchmarks as described here are more difficult to develop because they are not systematic (and they thus require more manpower). We provided several opportunities that we could consider in order to build such benchmarks.

[10]http://www.daml.org/ontologies/
[11]http://anhai.cs.uiuc.edu/archive/
[12]http://www.atl.external.lmco.com/projects/ontology/

Chapter 26

Automation

Both do2002as we carried out shown that the job of participants and of running the do2002a were greatly facilitated by providing tools for the do2002a. The tools also have the good features of providing the results to the participants without ambiguity.

We present below both what is already available and how it is desirable to develop tools for evaluating ontology alignment algorithms.

26.1 Test generation framework

We did not use so far any test generation system. However, our competence benchmark would highly benefit from such systematic test generation facility. It is thus necessary to have some tools which, from one ontology, are able to discard any of the features and to generate both the obtained ontology and the corresponding alignment.

This generation facility could be relatively easy to provide for simple changes such as discarding entities or replacing labels by random strings. It is a bit more complicated when it must:

- replace by missspellings which would require a missspelling generator;
- translate terms which would require an automatic translation tool (some could be used for that);
- flatten subsumption and composition hierarchies which is however feasible;
- expand subsumption and composition hierarchies in a meaningful way which is far more difficult.

Such a generation tool would take some ontology as input and systematically generate directories corresponding to the combination of all the features considered by the competence benchmark and containing the altered ontology plus the corresponding reference alignment.

It could be useful to implement such a tool with interactive manipulations.

26.2 Alignment framework

The I3CON Experiement Set Platform is a workbench under which the participants who wanted it could adapt their tools and plug them in for generating the results. It also provided formats in n3 notation for alignments and measures.

The EON Ontology Alignment Contest made use of the Alignment API[3] for representing the resulting alignments. This API provide many different services (see [Euzenat, 2004]).

For using the Alignment framework, do2002a participants have to implement the Alignment API. The Alignment API enables the integration of the algorithms based on a minimal interface. Adding new alignment algorithms amounts to create a new `AlignmentProcess` class implementing the interface. Generally, this class can extend the proposed `BasicAlignment` class. The `BasicAlignment` class defines the storage structures for ontologies and alignment specification as well as the methods for dealing with alignment display. All methods can be refined (no one is final). The only method it does not implement is the one that implement the alignment algorithm: `align`. This method is invoked from the `Alignment` object which is already connected with the ontologies. I takes a `Parameters` structure enabling to communicate the parameters to the algorithms and must fill the `Alignment` object with the correspondence `Cells` that have been found by the algorithm.

Once this class (which can be thought of as a wrapper around the alignment algorithm) is implemented, it is used by creating an alignment object, providing the two ontologies, calling the `align` method which takes parameters and initial alignment as arguments. The alignment object then bears the result of the alignment procedure. It is thus possible to invoke it on a particular set of tests with particular parameters and to output the results on a variety of formats.

This will be exploited by launching the `GroupAlign` facility of the Alignment API package to align all pairs of ontologies in a list of subdirectories and generate the result in the required format in this directory.

26.3 Evaluation framework

The do2002a framework must enable the comparison of an alignment with another one and to generate a resulting do2002a. One of the available methods of the Alignment API (`PRecEvaluator`) directly provides precision, recall and F-measure in an extension of the format developed by Lockheed Martin.

Since the contest, the tools around the API have been improved. The first improvement consists of comparing the results of different algorithms simultaneously and generating a table. Other developments will consist in providing the opportunity to directly launch an algorithm to a full test bench (and even to optimise some parameter). We will try to merge both tools.

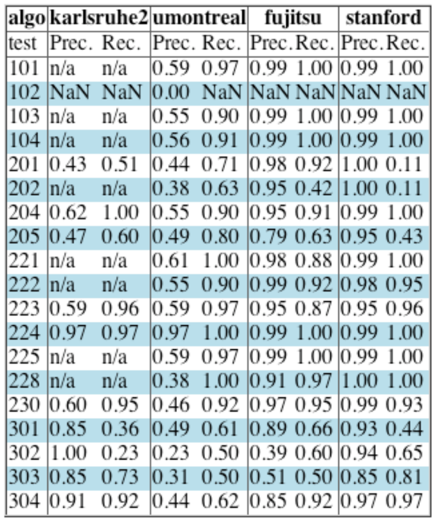
The do2002a framework is already implemented. It consists in gathering all the results in the same directory architecture and compare all of them to the reference alignment. This is implemented in the `GroupEval` class and has been used for the EON Ontology alignment contests (see Figure 26.1).

The Alignment API package provides a small utility (`GroupEval`) which allows to implement batch do2002a. It starts with a directory containing a set of subdirectories. Each subdirectory contains a reference alignment (usually called `refalign.rdf`) and a set of alignments (named `name1.rdf...namen.rdf`). These alignments can be provided directly by the `GroupAlign` facility.

Invoking `GroupEval` with the set of files to consider (-i argument) and the set of do2002a results to provide (-f argument with profm, for precision, recall, overall, fallout, f-measure as possible measures)

```
$ java -cp /Volumes/Phata/JAVA/ontoalign/lib/procalign.jar
    fr.inrialpes.exmo.align.util.GroupEval -f "pr" -c
    -l "karlsruhe2,umontreal,fujitsu,stanford"
```

returns an HTML file (which could also be other format) such as the one for Figure 26.1.



algo	karlsruhe2		umontreal		fujitsu		stanford	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
101	n/a	n/a	0.59	0.97	0.99	1.00	0.99	1.00
102	NaN	NaN	0.00	NaN	NaN	NaN	NaN	NaN
103	n/a	n/a	0.55	0.90	0.99	1.00	0.99	1.00
104	n/a	n/a	0.56	0.91	0.99	1.00	0.99	1.00
201	0.43	0.51	0.44	0.71	0.98	0.92	1.00	0.11
202	n/a	n/a	0.38	0.63	0.95	0.42	1.00	0.11
204	0.62	1.00	0.55	0.90	0.95	0.91	0.99	1.00
205	0.47	0.60	0.49	0.80	0.79	0.63	0.95	0.43
221	n/a	n/a	0.61	1.00	0.98	0.88	0.99	1.00
222	n/a	n/a	0.55	0.90	0.99	0.92	0.98	0.95
223	0.59	0.96	0.59	0.97	0.95	0.87	0.95	0.96
224	0.97	0.97	0.97	1.00	0.99	1.00	0.99	1.00
225	n/a	n/a	0.59	0.97	0.99	1.00	0.99	1.00
228	n/a	n/a	0.38	1.00	0.91	0.97	1.00	1.00
230	0.60	0.95	0.46	0.92	0.97	0.95	0.99	0.93
301	0.85	0.36	0.49	0.61	0.89	0.66	0.93	0.44
302	1.00	0.23	0.23	0.50	0.39	0.60	0.94	0.65
303	0.85	0.73	0.31	0.50	0.51	0.50	0.85	0.81
304	0.91	0.92	0.44	0.62	0.85	0.92	0.97	0.97

Figure 26.1: Precision and recall results for various alignment algoritms in HTML format.

26.4 Conclusion

Providing formats has the advantage of being able to compute new measures when the consensus is latter made on a new do2002a method. The set of tools that we have presented help automating the generation of tests and do2002a of results. As can be seen from Figure 26.2, the three presented functions should be able to automate generation, processing and do2002a of the alignment algorithm f on the basis of ontology O. This has the advantage of decreasing the rate of errors and with it the risk of complaints. This also lowers the costs of generating a new set of tests or evaluating again the set of results. This helps managing the do2002as. But these tools also reduces the amount of work necessary from the participants to run the tests. They can thus concentrate on performing at best. Moreover, automation enables the participants to run the do2002a of their results easily which helps them to report problems early and to improve their algorithms against the actual benchmark results.

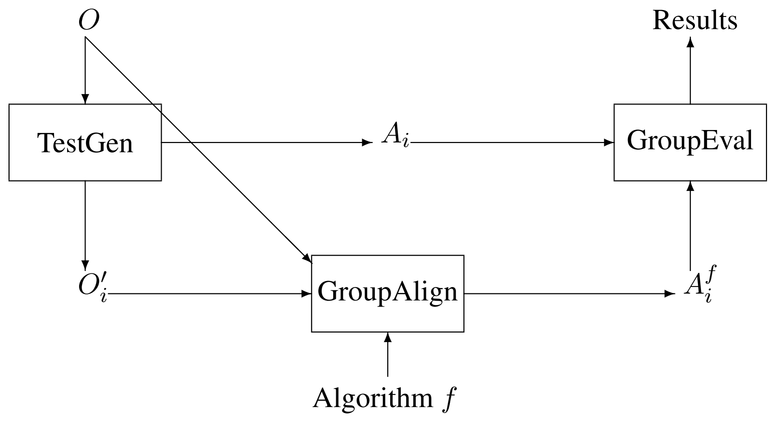


Figure 26.2: Process flow provided by the tool suite.

Conclusion

We started from the goals of the do2002a (helping improving the state of the art in ontology alignment technology) and the definition of the alignment task. From this, and our experience of running and participating in two do2002a campaigns during the first year of Knowledge web, we have designed what we think should be a practical specification of such benchmarks.

It is based on a recurring yearly event combining the processing of a set of competence benchmark tests helping characterising the behaviour of each algorithm and a performance benchmark aiming at comparing algorithms performances on real world ontologies.

Our task in the coming months will consist in instantiating this framework and organising the second benchmarking campaign.

These benchmarks could, of course, also be useful in selecting the most appropriate tool for a particular task or application.

Appendix: Evaluation experience

We present two different experiments that recently occured:

– The Information Interpretation and Integration Conference (I3CON), to be held at the NIST Performance Metrics for Intelligent Systems (PerMIS) Workshop, will be an ontology alignment demonstration competition on the model of the NIST Text Retrieval Conference. This contest focuses on "real-life" test cases and compare algorithm global performance. This effort has mainly been developed by Todd Hughes and Benjamin Ashpole of Lockheed Martin.
– The Ontology Alignment Contest at the 3rd Evaluation of Ontology-based Tools (EON) Workshop, to be held the International Semantic Web Conference (ISWC), will target the characterization of alignment methods with regard to particular ontology features. This initiative aims at defining a proper set of benchmark tests for assessing feature-related behavior. Because of its emphasis on evaluating the performances of tools instead of the competition between them, the term contest was not the best one. This contests has mostly been designed by Jrme Euzenat at INRIA.

There was two different initiatives because the idea of evaluating alignment methods had been out since a long time [Noy and Musen, 2002a; Do *et al.*, 2002] and there had been two occasions at once.

First competitive do2002a (I3CON)

The I3CON[1] has been designed for providing some do2002a of ontology alignment algorithms.

The do2002a methodology consisted in publishing a set of ontologies to be compared with another ontology. As in the other test, the participants were asked to run one tool in one configuration on all the tests and to provide the results in a particular format. This format being similar but different from the previous one.

Contrary to the EON Ontology Alignment Contest, no reference alignment was provided, so the participant could not tune their system to find the best results for these tests.

A training set of two ontology pairs with their hand-made reference alignments was provided to the potential participant before the actual test cases so that they could adapt their systems for the contest.

The do2002a measure used were as usual: precision, recall and f-measure with regard to one secret reference alignment. No performance time measures were required.

A set of tool for running the tests was provided.

[1] `http://www.atl.external.lmco.com/projects/ontology/i3con.html`

Test set

The set of tests was made of 8 ontology pairs. The ontologies concerned various domains (animals, Russia, soccer, basket ball, hotels, networks). The initial idea of the contest was to find ontology pairs on the web. However, this was not easy, so the organisers ended up by taking ontologies on the web and altering them. Various techniques have been used for the alteration (from random to adapting other ontologies concerning a related topic or using language translation).

The ontologies were provided in RDF/XML and n3, but their ontology language could be RDFS, DAML+OIL or OWL.

All the ontologies and reference alignments were produced by hand by consensus of an external team of students.

Lesson learned from the first competition

The first good news of the I3CON experiment is that such a competition is possible.

For both do2002as, we expected five participants. There where five teams entering the I3CON initiative (ATL/Lockheed Martin, AT&T, INRIA, Karlsruhe and Teknowledge). This result is not too bad for a first run of experiment.

The results do not show a clear winner, some tests are more difficult to some algorithms than others. It is quite difficult to learn from the results of this experiments, and this reinforce the benefit of having papers from the participants.

The performance of the experiment itself showed that it was quite difficult to find pair of ontologies suitable for such a benchmark: either they are not comensurate, or they are expressed in different languages or the alignment does not exist. The ontology pairs have thus been, for most of them created for the purposes of the do2002a on a realistic basis.

Contrary to the EON Ontology Alignment Contest, participants did not use the tools provided to them. This could come from a lack of documentation, and should be investigated for further efforts.

First competence benchmarks (EON Ontology Alignment Contest)

The EON "Ontology alignment contest"[2] has been designed for providing some do2002a of ontology alignment algorithms.

The do2002a methodology consisted in publishing a set of ontologies to be compared with another ontology. The participants were asked to run one tool in one configuration on all the tests and to provide the results in a particular format. In this format[3], an alignment is a set of pairs of entities from the ontologies, a relation supposed to hold between these entities and a confidence measure in the aligned pair. The tools could use any kind of available resources, but human intervention.

Along with the ontologies, a reference alignment was provided (in the same format). This alignment is the target alignment that the tools are expected to find. The reference alignment has all its confidence measures to the value 1 and most of the relations were equivalence (with very few subsumption relations). Because of the way the tests have been designed (see below), these alignments should not be contested. The participant were allowed to compare their results to the

[2]http://co4.inrialpes.fr/align/Contest
[3]http://www.inrialpes.fr/exmo/software/ontoalign/

output of their systems and the reference alignment and to chose the best tuning of their tools (overall).

The full test bench was proposed for examination to potential participants for 15 days prior to the final version. This allowed participants to provide some comments that could be corrected beforehand. Unfortunately, the real comments came later.

The results of the tests were expected to be given in terms of precision and recall of correspondences found in the produced alignment compared to the reference alignment. No performance time measures were required. The participants were also asked to provide a paper, in a predefined format, describing their tools, their results and comments on the tests.

Tools were provided for manipulating the alignments and evaluate their precision, recall and other measures[3].

Test set

The set of tests consisted in one medium ontology (33 named classes, 39 object properties, 20 data properties, 56 named individuals and 20 anonymous individuals) to be compared to other ontologies. All ontologies were provided in OWL under its RDF/XML format.

This initial ontology was about a very narrow domain (bibliographical references). It was designed by hand from two previous efforts. It took advantage of other resources whenever they were available. To that extent the reference ontology refers to the FOAF (Friend-of-a-friend) ontology and the iCalendar ontology.

There were three series of tests:

– simple tests such as comparing the reference ontology with itself, with another irrelevant ontology (the wine ontology used in the OWL primer) or the same ontology in its restriction to OWL-Lite;
– systematic tests that were obtained by discarding some features of the initial ontology leaving the remainder untouched. The considered features were (names, comments, hierarchy, instances, relations, restrictions, etc.). This approach aimed at recognising what tools really need. Our initial goal was to propose not just one feature discard but all the combinations of such. Unfortunately, we were unable to provide them before the launch of the contest.
– four real-life ontologies of bibliographic references that were found on the web and left untouched.

All the ontologies and reference alignments were produced by hand in a very short time. This caused a number of problems in the initial test base that were corrected later.

Results

The results of the EON contest [Sure *et al.*, 2004] were globally higher than these of the I3CON certainly because of the way benchmark were made (all coming from the same source) with very identified and localised distortion.

As a first note, we expected five participants but finally only four entered (Stanford/SMI, Fujitsu, INRIA & UoMontral and Karlsruhe). This is few, especially with regard to all the alignments algorithms out there. We hope that these four participants are the pionneer who will induce the others to put their work under comparison.

algo	karlsruhe2		umontreal		fujitsu		stanford	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
101	n/a	n/a	0.59	0.97	0.99	1.00	0.99	1.00
102	NaN	NaN	0.00	NaN	NaN	NaN	NaN	NaN
103	n/a	n/a	0.55	0.90	0.99	1.00	0.99	1.00
104	n/a	n/a	0.56	0.91	0.99	1.00	0.99	1.00
201	0.43	0.51	0.44	0.71	0.98	0.92	1.00	0.11
202	n/a	n/a	0.38	0.63	0.95	0.42	1.00	0.11
204	0.62	1.00	0.55	0.90	0.95	0.91	0.99	1.00
205	0.47	0.60	0.49	0.80	0.79	0.63	0.95	0.43
221	n/a	n/a	0.61	1.00	0.98	0.88	0.99	1.00
222	n/a	n/a	0.55	0.90	0.99	0.92	0.98	0.95
223	0.59	0.96	0.59	0.97	0.95	0.87	0.95	0.96
224	0.97	0.97	0.97	1.00	0.99	1.00	0.99	1.00
225	n/a	n/a	0.59	0.97	0.99	1.00	0.99	1.00
228	n/a	n/a	0.38	1.00	0.91	0.97	1.00	1.00
230	0.60	0.95	0.46	0.92	0.97	0.95	0.99	0.93
301	0.85	0.36	0.49	0.61	0.89	0.66	0.93	0.44
302	1.00	0.23	0.23	0.50	0.39	0.60	0.94	0.65
303	0.85	0.73	0.31	0.50	0.51	0.50	0.85	0.81
304	0.91	0.92	0.44	0.62	0.85	0.92	0.97	0.97

Table 26.1: Precision and recall results for each test

Below is the table of precision and recall results computed on the output provided by the participants with the help of the alignment API implementation.

Here are some consideration of the results obtained by the various participants. These are not statistically backed up and only corresponds to a rough analysis. More explanations are found in the papers presented by the participants.

There were two groups of competitors...

In this test, there are clear winners it seems that the results provided by Stanford and Fujitsu/Tokyo outperform those provided by Karlsruhe and Montréal/INRIA.

In fact, it can be considered that these constitute two groups of programs. The Stanford+Fujitsu programs are very different but strongly based on the labels attached to entities. For that reason they performed especially well when labels were preserved (i.e., most of the time). The Karlsruhe+INRIA systems tend to rely on many different features and thus to balance the influence of individual features, so they tend to reduce the fact that labels were preserved[4].

This intuition should be further considered in the light of more systematic tests which were planned but never made.

...and indeed three groups of tests

The results[5] of the I3CON inititative were relatively homogeneous in the sense that no algorithm was clearly outperforming the others in all tasks and no task what more difficult than others.

Without going through a throughout statistical analysis of the results, it seems that the separation between three sets of test that we presented (indicated by the first digit of their numbers) is significant for the participants as well.

- The first four tests were relatively easily handled by all participants. All programs showed there a better recall than precision (but not very significant).
- The systematic tests show more difficulty for the programs in general and for those of the second group in particular.
- The real life test were even more dificicult for both groups of participants.

Lesson learned from the first benchmarking campain

The first good thing that we learnt is that it is indeed possible to run such a test.

Another lesson that we have learnt is that OWL is not that homogeneous when tools have to manipulate it. Parsers and API for OWL (e.g., Jena and OWL-API) are not really aligned in their way to handle OWL ontologies. This can be related to very small matters which can indeed render difficult entering the challenge. It is our expectation that these products will improve in the coming year. For the moment we modified the files in order to avoid these problems.

It is very difficult indeed to synthesize these results. This is true because there were different tests and not all go in the same direction. So aggregating the results can be done in many ways (e.g., averaging, global P/R, counting dominance). Moreover, these results are based on two measures, precision and recall, which are very easily understood but dual in the sense that increasing one often decreases the other. This means that one algorithms can have sometimes the same results

[4]It seems also that these two programs produced some artefact that should be easily eliminated

[5]`http://www.atl.external.lmco.com/projects/ontology/papers/I3CON-Results.pdf`

algo	karlsruhe2		umontreal		fujitsu		stanford	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	.	.	0,57	0,93	0,99	1,00	0,99	1,00
2xx	0,61	0,83	0,55	0,89	0,95	0,86	0,98	0,77
3xx	0,90	0,56	0,37	0,56	0,66	0,67	0,92	0,72
total	0,73	0,72	0,51	0,82	0,89	0,84	0,97	0,80

Table 26.2: Average value of precision recall per groups of tests and globally

as another but they are found non comparable in the table. As an indication, the average values are given in Figure 26.2.

It is noteworthy that for solving this problem, TREC uses curves on the precision×recall space. But we do not now yet what is the protocol for obtaining such curves (may be by asking some ranking of the answers).

People appreciated to be given tools to manipulate the required formats. It is clear that in order to attract participants, the test process should be easy.

We also realised that the production of an incomplete test bench (not proposing all combinations of discarded features) had an influence on the result. As a matter of fact, algorithms working on one feature only were advantaged because in most of the tests this feature was preserved.

Another lesson we learned is that asking for a detailed paper was a very good idea. We have been pleased of how much insight can be found in the comments of the competitors.

Chapter 27

Generation of Reference Mappings

One of the problem we are faced when designing test cases for do2002a is that of acquiring the reference alignments. Up to now the acquisition of the reference mappings that hold among the nodes of two taxonomies is performed manually. Similarly to the annotated corpora for information retrieval or information extraction, a corpus of pairwise relationships is annotated. Of course such an approach prevents the opportunity of having large corpora. The number of mappings between two taxonomies are quadratic with respect to taxonomy size, what makes hardly possible the manual mapping of real world size taxonomies. It is worthwhile to remember that web directories, for example, have tens thousands of nodes. Certain heuristics can help in reducing the search space but the human effort is still too demanding. This is our goal here to provide such a method in order to design decent test sets.

Our proposal is to build a reference interpretation for a node looking at its use. We argue that the semantics of nodes can be derived by their pragmatics, i.e., how they are used. In context of web directories, the nodes of a taxonomy are used to classify documents. The set of documents classified under a given node implicitly defines its meaning. This approach has been followed by other researchers. For example in [Doan *et al.*, 2003a; Ichise *et al.*, 2003] the interpretation of a node is approximated by a model computed through statistical learning. Of course the accuracy of the interpretation is affected by the error of the learning model. We follow a similar approach but without the statistical approximation. Our working hypothesis is that the meaning of two nodes is equivalent if the sets of documents classified under those nodes have a meaningful overlap. The basic idea is to compute the relationship hypotheses based on the co-occurence of documents. This document-driven interpretation can be used as a reference value for the do2002a of competing matching solutions. A simple definition of equivalence relationship based on documents can be derived by the F1 measure of information retrieval.

Figure 27.1 shows a simple example. In the graphical representation we have two taxonomies, for each of them we focus our attention on a reference node. Let be S and P two sets of documents classified under the reference nodes of the first and second taxonomies respectively. We will refer to A_S and A_P as the set of documents classified under the ancestor nodes of S and P. Conversely, we will refer to T_S and T_P as the set of documents classified under the subtrees of S and P. The goal is to define a relationship hypothesis based on the overlapping of the set of documents, i.e. the pragmatic use of the nodes.

The first step, the *equivalence* relationship, can be easily formulated as the F1 measure of information retrieval [Baeza-Yates and Ribeiro-Neto, 1999]. The similarity of two sets of documents is defined as the ratio between the marginal sets and the shared documents:

$$Equivalence = \frac{|O_P^S|}{|M_P^S| + |M_S^P|}$$

where the set of shared documents is defined as $O_P^S = P \cap S$ and $M_P^S = S \setminus O_P^S$ is the marginal set of documents classified by S and not classified by P (similarly $M_S^P = P \setminus O_P^S$). The following equivalence applies $O_P^S = O_S^P$. Notice that "O" stands for "overlapping" and "M" stands for "Marginal set".

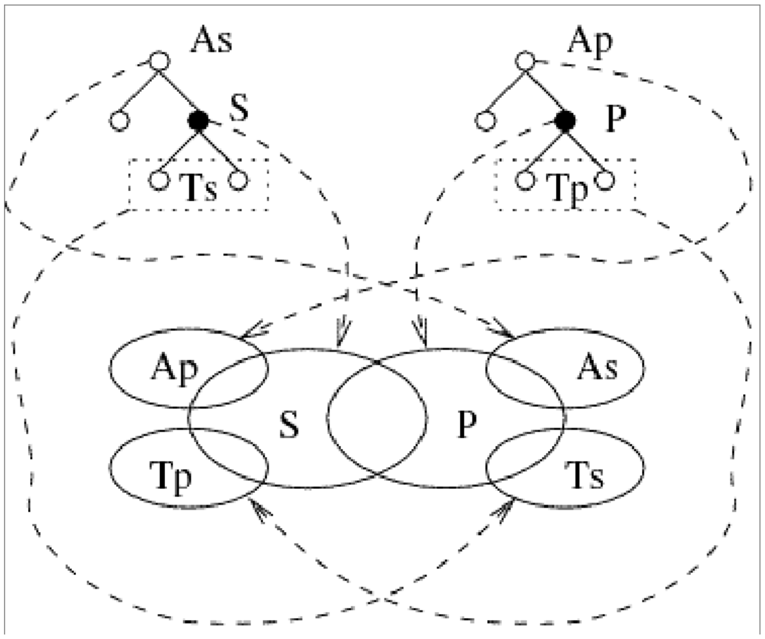


Figure 27.1: The pairwise relationships between two taxonomies.

We do a step forward because we do not only compute the equivalence hypothesis based on the notion of F1 measure of information retrieval, but we extend such equation to define the formulation of generalization and specialization hypotheses. Generalization and specialization hypotheses can be formulated taking advantage of the contextual encoding of knowledge in terms of hierarchies of categories. The challenge is to formulate a generalization hypothesis (and conversely a specialization hypothesis) between two nodes looking at the overlapping of set of documents classified in the ancestor or subtree of the reference nodes [Avesani, 2002].

The *generalization* relationship holds when the first node has to be considered more general than the second node. Intuitively, it happens when the documents classified under the first nodes occur in the ancestor of the second node, or the documents classified under the second node occur in the subtree of the first node. Following this intuition we can formalize the generalization hypothesis as

$$Generalization = \frac{|O_P^S| + |O_{A_S}^P| + |O_{T_P}^S|}{|M_P^S| + |M_S^P|}$$

where $O_{A_S}^P$ represents the set of documents resulting from the intersection between M_S^P and the set of documents classified under the concepts in the hierarchy above S (i.e. the ancestors); similarly $O_{T_P}^S$ represents the set of documents resulting from the intersection between M_P^S and the set of documents classified under the concepts in the hierarchy below P (i.e. the children).

In a similar way we can design the *specialization* relationship. The first node is more specific than the second node when the meaning associated to the first node can be subsumed by the meaning of the second node. Intuitively, this happens when the documents classified under the first nodes occur in the subtree of the second node, or the documents classified under the second node occur in the ancestor of the first node.

$$Specialization = \frac{|O_P^S| + |O_{T_S}^P| + |O_{A_P}^S|}{|M_P^S| + |M_S^P|}$$

where $O_{T_S}^P$ represents the set of documents resulting from the intersection between M_S^P and the set of documents classified under the concepts in the hierarchy below S (i.e. the children); similarly $O_{A_P}^S$ represents the set of documents resulting from the intersection between M_P^S and the set of documents classified under the concepts in the hierarchy above P (i.e. the ancestors).

The three definitions above allow us to compute a relationship hypothesis between two nodes of two different taxonomies. Such an hypothesis relies on the assumption that if two nodes classify the same set of documents, the meaning associated to the nodes is reasonably the same. Of course this assumption is true for a virtually infinite set of documents. In a real world case study we face with finite set of documents, and therefore, this way of proceeding is prone to error. Nevertheless, our claim is that the approximation introduced by our assumption is balanced by the benefit of scaling with the annotation of large taxonomies.

27.1 Classification Hierarchies

Let us try to apply the notion of document-driven interpretation to a real world case study. We focus our attention to web directories for many reasons. Web directories are widely used and known; moreover they are homogeneous, that is they cover general topics. The meaning of a node in a web directory is not defined with formal semantics but by pragmatics. Furthermore the web directories address the same space of documents, therefore the working hypothesis of co-occurence of documents can be sustainable. Of course different web directories don't cover the same portion of the web but the overlapping is meaningful. The case study of web directories meets two requirements of the matching problem: to have heterogeneous representations of the same topics and to have taxonomies of large dimensions.

We address three main web directories: Google, Yahoo! and Looksmart. Nodes have been considered as categories denoted by the lexical labels, the tree structures have been considered as hierarchical relations, and the URL classified under a given node as documents. The following table summarizes the total amount of processed data.

Web Directories	Google	Looksmart	Yahoo!
number of nodes	335.902	884.406	321.585
number of urls	2.425.215	8.498.157	872.410

Let us briefly describe the process by which we have arranged an annotated corpus of pairwise relations between web directories.

Step 1. We crawled all three web directories, both the hierarchical structure and the web contents, then we computed the subset of URLs classified by all of them.

Step 2. We pruned the downloaded web directories by removing all the URLs that were not referred by all the three web directories.

Step 3. We performed an additional pruning by removing all the nodes with a number of URLs under a given threshold. In our case study we fixed such a threshold at 10.

Step 4. We manually recognized potential overlapping between two branches of two different web directories like

```
   Google:/Top/Science/Biology
Looksmart:/Top/Science-and-Health/Biology

    Yahoo:/Top/Computers-and-Internet/Internet
Looksmart:/Top/Computing/Internet

   Google:/Top/Reference/Education
    Yahoo:/Top/Education
```

We recognized 50 potential overlapping and for each of them we run an exhaustive assessment on all the possible pairs between the two related subtrees. Such an heuristic allowed us to reduce the quadratic explosion of cartesian product of two web directories. We focussed the analysis on smaller subtrees where the overlaps were more likely.

Step 5. We computed the three document-driven hypothesis for *equivalence*, *generalization* and *specialization* relationships as described above. Hypotheses of equivalence, generalization and specialization are normalized and estimated by a number in the range [0,1]. Since the cumulative hypothesis of all three relationships for the same pair of nodes can not be higher than 1, we introduce a threshold to select the winning hypothesis. We fixed such a threshold to 0.5.

We discarded all the pairs where none of the three relationship hypotheses was detected. This process allowed us to obtain 2265 pairwise relationships defined using the document-driven interpretation. Half are equivalence relationships and half are generalization relationships (notice that by definition generalization and specialization hypothesis are symmetric).

27.2 Thesauri and Ontologies

Validation of the results on the medical ontologies matching task is still an open problem. The results can be replicated in straightforward way. At the same time there are no sufficiently big set of the reference mappings what makes impossible calculation of the matching quality measures. In contrast to the generation of reference alignments for classification hierachies, we do not assume that instance data is available or that the models are represented in the same way or using the same language. Normally, the models will be from the same domain (eg. medicine or business). The methodology consists of four basic steps. In the first step, basic decisions are made about the representation of the conceptual models and instance data to be used. In the second step instance data is created by selecting it from an existing set or by classifying data according to the models under consideration. In the third step, the generated instance data is used to generate candidate mappings based on shared instances. In the forth step finally, the candidate mappings are evaluated against a set of quality criteria and the final set of reference mappings is determined.

Step 1. Preparation

The first step of the process is concerned with data preparation. In particular, we have to transform the conceptual models into a graph representation and select and prepare the appropriate instance data to be used to analyze overlap between concepts in the different models. We structure this step based on the KDD process for Knowledge Discovery and Data Mining.

Step 2. Instance Classification

In the second step the chosen instance data is classified according to the different conceptual models. For this purpose, an appropriate classification method has to be chosen that fits the data and the conceptual model. Further, the result of the classification process has to be evaluated. For this step we rely on established methods from Machine Learning and Data Mining.

Step 3. Hypothesis Generation

In the third step, we generate hypothesis for reference mappings based on shared instances created in the first two steps. In this step, we prune the classification by removing instances that are classified with a low confidence and selecting subsets of the conceptual models that show sufficient overlap. We further compute a degree of overlap between concepts in the different models and based on this degree of overlap select a set of reference mappings between concepts with a significant overlap.

Step 4. Evaluation

In the last step, the generated reference mapping is evaluated against the result of different matching systems as described in [Avesani *et al.*, 2005] using a number of criteria for a reference mapping. These criteria include correctness, complexity of the mapping problem and the ability of the mappings to discriminate between different matching approaches.

We are testing this methodology using a data set of medical documents called OHSUMED. The data set contains 350.000 articles from medical journals covering all aspects of medicine.

For classifying these documents according to the two ontologies of anatomy, we use the Collexis text indexing and retrieval system that implements a number of automatic methods for assigning concepts to documents. Currently, we are testing the data set and the system on a subset of UMLS with known mappings in order to assess the suitability of the methodology. The generation of the reference mappings for the Anatomy case will proceed around the end of 2005 and we are hopeful to have thoroughly tested set of reference mappings for the 2006 alignment challenge.

27.3 Evaluation Results

The do2002a was designed in order to assess the major dataset properties namely:

– *Complexity*, namely the fact that the dataset is ”hard” for state of the art matching systems.

– *Discrimination ability*, namely the fact that the dataset can discriminate among various matching approaches.

– *Incrementality*, namely the fact that the dataset allows to incrementally discover the weaknesses of the tested systems.

– *Correctness*, namely the fact that the dataset can be a source of correct results.

We have evaluated two state of the art matching systems *COMA* and $S - Match$ and compared their results with *baseline solution*. Let us describe the matching systems in more detail.

The *COMA* system [Do and Rahm, 2002] is a generic syntactic schema matching tool. It exploits both element and structure level techniques and combines the results of their independent execution using several aggregation strategies. *COMA* provides an extensible library of matching algorithms and a framework for combining obtained results. Matching library contains 6 individual matchers, 5 hybrid matchers and 1 reuse-oriented matcher. One of the distinct features of the COMA tool is the possibility of performing iterations in the matching process. In the do2002a we used default combination of matchers and aggregation strategy (*NamePath+Leaves* and *Average* respectively).

S-Match is a generic semantic matching tool. It takes two tree-like structures and produces a set of mappings between their nodes. *S-Match* implements semantic matching algorithm in 4 steps. On the first step the labels of nodes are linguistically preprocessed and their meanings are obtained from the Oracle (in the current version WordNet 2.0 is used as an Oracle). On the second step the meaning of the nodes is refined with respect to the tree structure. On the third step the semantic relations between the labels at nodes and their meanings are computed by the library of element level semantic matchers. On the fourth step the matching results are produced by reduction of the node matching problem into propositional validity problem, which is efficiently solved by SAT solver or ad hoc algorithm (see [Giunchiglia *et al.*, 2004; Giunchiglia *et al.*, 2005] for more details).

We have compared the performance of these two systems with *baseline solution*. It is executed for each pair of nodes in two trees. The algorithm considers a simple string comparison among

Table 27.1: Evaluation Results

	Google vs. Looksmart	Google vs. Yahoo	Looksmart vs.Yahoo	Total
COMA	608	250	18	876 (38,68%)
$=$	608	250	18	876
$\subseteq$	N/A	N/A	N/A	N/A
$\supseteq$	N/A	N/A	N/A	N/A
S-Match	584	83	2	669 (29,54%)
$=$	2	5	0	7
$\subseteq$	46	19	2	67
$\supseteq$	536	59	0	595
Baseline	54	76	0	130 (5,39%)
$=$	52	0	0	52
$\subseteq$	0	76	0	76
$\supseteq$	2	0	0	2

the labels placed on the path spanning from a node to the root of the tree. Equivalence, more general and less general relations are computed as the corresponding logical operations on the sets of the labels.

The systems have been evaluated on the dataset described in Section 30.2.1. We computed the number of matching tasks solved by each matching system. Notice that the matching task was considered to be solved in the case when the matching system produce specification, generalization or equivalence semantic relation for it. For example, TaxME suggests that specification relation holds in the following example:

```
Google:/Top/Sports/Basketball/Professional/NBDL
Looksmart:/Top/Sports/Basketball
```

COMA produced for this matching task 0.58 similarity coefficient, which can be considered as equivalence relation with probability 0.58. In the do2002a we consider this case as true positive for *COMA* (i.e., the mapping was considered as found by the system).

Notice that at present TaxME contains only true positive mappings. This fact allows to obtain the correct results for Recall measure, which is defined as a ratio of reference mappings found by the system to the number of reference mappings. At the same time Precision, which is defined as ratio of reference mappings found by the system to the number of mappings in the result, can not be correctly estimated by the dataset since, as from Section 30.2.1, TaxME guarantee only the correctness but not completeness of the mappings it contains.

Evaluation results are presented on Table 27.1. It contains the total number of mappings found by the systems and the partitioning of the mappings on semantic relations. Let us discuss the results through the major dataset properties perspective.

27.3.1 Complexity

As from Table 27.1, the results of *baseline* are surprisingly low. It produced slightly more than 5% of mappings. This result is interesting since on the previously evaluated datasets (see [Bouquet *et al.*, 2003c] for example) the similar baseline algorithm performed quite well and found up to 70%

of mappings. This lead us to conclusion that the dataset is not trivial (i.e., it is essentially hard for simple matching techniques).

As from Figure 27.2, *S-Match* found about 30% of the mappings in the biggest (Google-Yahoo) matching task. At the same time it produced slightly less than 30% of mappings in all the tasks. *COMA* found about 35% of mappings on Google-Looksmart and Yahoo-Looksmart matching tasks. At the same time it produced the best result on Google-Yahoo. *COMA* found slightly less than 40% of all the mappings. These results are interesting since, as from [Do and Rahm, 2002; Giunchiglia *et al.*, 2004], previously reported recall values for both systems were in 70-80% range. This fact turn us to conclusion that the dataset is hard for state of the art syntactic and semantic matching systems.

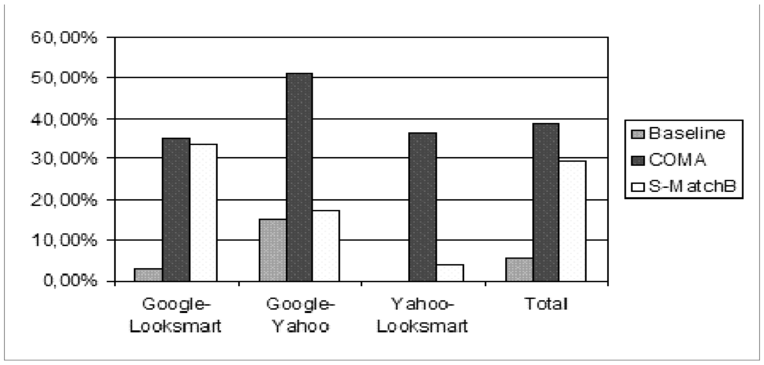


Figure 27.2: Percentage of correctly determined mappings(Recall)

27.3.2 Discrimination ability

Consider Figure 27.3. It presents the partitioning of the mappings found by *S-Match* and *COMA*. As from the figure the sets of mappings produced by *COMA* and *S-Match* intersects only on 15% of the mappings. This fact turns us to an important conclusion: the dataset is discriminating (i.e., it contains a number of features which are essentially hard for various classes of matching systems and allow to discriminate between the major qualities of the systems).

Figure 27.3: Partitioning of the mappings found by COMA and S-Match

27.3.3 Incrementality

In order to evaluate incrementality we have chosen *S-Match* as a test system. In order to identify the shortcomings of *S-Match* we manually analyzed the mappings missed by *S-Match*. This analysis allowed us to clasterize the mismatches into several categories. We describe in detail one of the most important categories of mismatches namely *Meaningless labels*.

Consider the following example:

```
Google:/Top/Science/Social_Sciences/Archaeology/Alternative/
       South_America/Nazca_Lines
Looksmart:/Top/Science_&_Health/Social_Science/Archaeology/
       By_Region/Andes_South_America/Nazca
```

In this matching task some labels are meaningful in the sense they define the context of the concept. In our example these are *Social_Sciences*, *Archaeology*, *South_America*, *Nazca*. The other labels do not have a great influence on the meaning of concept. At the same time they can prevent *S-Match* from producing the correct semantic relation. In our example *S-Match* can not find any semantic relation connecting *Nazca_Lines* and *Nazca*. The reason for this is *By_Region* label, which is meaningless in the sense it is defined only for readability and taxonomy partitioning purposes. An other example of this kind is

```
Google:/Top/Arts/Celebrities/A/Affleck,_Ben
Looksmart:/Top/Entertainment/Celebrities/Actors/Actors_A/
        Actors_Aa-Af/Affleck,_Ben/Fan_Dedications
```

Here, *A* and *Actors_A/Actors_Aa-Af* do not influence on the meaning of the concept. At the same time they prevent *S-Match* to produce the correct semantic relation holding between the concepts.

An optimized version of *S-Match* (*S-Match*++) has a list of meaningless labels. At the moment the list contains only about 30 words but it is automatically enriched in preprocessing phase. A general rule for considering natural language label as meaningless is to check whether it is used for taxonomy partitioning purposes. For example, *S-Match*++ consider as meaningless the labels with the following structure *by* $\langle word \rangle$, where $\langle word \rangle$ stands for any word in natural language. However, this method is not effective in the case of labels composed from alphabet letters (such as *Actors_Aa-Af* from previous example). *S-Match*++ deals with the latter case in the following way: the combination of letters are considered as meaningless if it is not recognized by WordNet, not in abbreviation or proper name list, and at the same time its length is less or equal to 3. The addition of these techniques allowed to improve significantly the *S-Match* matching capability. The number of mappings found by the system on TaxME dataset increased by 15%. This result gives us an evidence to incrementality of the dataset (i.e., the dataset allows to discover the weaknesses of the systems and gives the clues to the systems evolution).

Analysis of *S-Match* results on TaxME allowed to identify 10 major bottlenecks in the system implementation. At the moment we are developing ad hoc techniques allowing to improve *S-Match* results in this cases. The current version of *S-Match* (*S-Match*++) contains the techniques allowing to solve 5 out of 10 major categories of mismatches. Consider Figure **??**.

27.3.4 Correctness

We manually analyzed correctness of the mappings provided by TaxME. At the moment 60% of mappings are processed and only 2-3% of them are not correct. Taking into account the notion of idiosyncratic classification [Goren-Bar and T.Kuflik, 2005] (or the fact that human annotators on the sufficiently big and complex dataset tend to have resemblance up to 20% in comparison with their own results), such a mismatch can be considered as marginal.

Chapter 28

Alternative tracks

We propose here a number of possible new tracks for next do2002a campaign. Their goal is to evaluate differently existing systems or to evaluate other features of the systems. One obvious feature that still has to be investigated is the time taken by algorithms.

28.1 Unconstrained discovery scenario

The currently used scenario of alignment do2002a is that of a contest. Its rules encourage the participants to seek maximal quality of individual alignments, which can be subsequently evaluated in terms of precision/recall with respect to a "golden standard" results (defined a priori but hidden from the participants). However, focussing on numerically measures of quality only is somewhat limiting wrt. the scope of observations potentially produced by automated alignment tools: more sophisticated observations, which can give interesting insight into the nature of tools as well as that of data but cannot be evaluated by traditional metrics, could arise. Let us consider, for example, alignments of contiguous paths in a tree, or "crossed alignments" that invert the taxonomic relationship. For ontologies with axioms (see the newly introduced notion of "Parallel OWL-DL ontologies"), even the logical difference in definitions could be explicitly captured. Sharing such heterogeneous observations clearly goes beyond the "contest" scenario of do2002a, as they can only be evaluated for subjective "interestingness".

There is an analogy with similar do2002a activities in the more traditional research area of Knowledge Discovery in Databases; discovery of ontological alignments can indeed be viewed as special case of knowledge discovery ("data mining"). The KDD Cup[1] enforces the type of analysis to be performed on the given dataset, compares the results obtained by different tools with correct results known a priori, yields a ranking, and, finally, awards the winner. On the other hand, in the Discovery Challenge [2], no correct results are known in advance: the researchers analyze the same data in different ways and with different methods, and share observations about their heterogeneous results (within a dedicated workshop).

The main advantage of this approach is the relatively unrestricted range of tasks to be carried out on data, and even the possibility to publish and discuss negative results, which may often be as

[1] `http://kdd05.lac.uic.edu/kddcup.html`
[2] `http://lisp.vse.cz/challenge`

useful as positive ones. A possible different track for future editions of the Ontology Alignment Evaluation Initiative could be an open workshop for different participants to contribute discussing and "negociating" the alignments. If some consensus is made, this can be further used as golden standards for other experiments.

28.2 Parallel OWL-DL ontologies

Within a recently launched informal initiative nicknamed OntoFarm, a new collection is being built by joint effort of multiple independent contributors from European research institutes (within as well as outside Knowledge Web). The chosen domain is that of conference organisation, including both programme and technical matters. To date, a pilot ontology (with about 50 concepts, 30 properties, and numerous axioms) derived from the structure of the EKAW conference) exist, and about 4-5 other are envisaged to arise in early Spring 2006. Due to its following characteristics, the new collection should improve on the tests cases provided in previous issues.

Richness in OWL-DL Constructs Most existing alignment tests focus on taxonomies of terms. However, many semantic web application scenarios assume complex ontologies that allow non-trivial reasoning. The design of the new collection will explicitly address the inclusion of full OWL-DL axiom types.

Larger size of collection Most existing alignment tests are limited to a pair of ontologies only. Here we consider multiple ontologies describing the same domain. This enables to consider more complex (meta-)alignment patterns, for example, such that some matching segments from two ontologies do not have match in the third one.

Interpretability by Non-Experts Despite Real-World flavor Complex real-world ontologies (such as those from the bio/medical domain) require a domain expert to properly interpret their concepts, while knowledge engineers can only handle them at the technical level. Here we intentionally chose a domain that is perfectly understandable (at least) to any researcher. On the other hand, by our experience with building the pilot ontology, the domain is non-trivial, shares many aspects with heavier-weighted industrial activities, and can give rise to numerous concepts, properties and axioms. Each ontology will model the domain of conference organization from the point of view of a concrete conference series its developer is deeply involved with. We thus believe that the collection, itself being "artificial" (i.e. created on purpose), will have heterogeneity introduced in a natural way, and its analysis will thus mimic real-life semantic web scenarios reasonably well.

Availability of Instance Data While in applications like business or medical applications, real instance data are subject to strong privacy constraints, data about organizers, committees, authors, presenters etc. of conferences are typically public and can even be picked-up from websites with reasonable effort. Information Extraction and Wrapper technology (also developed at UEP) can serve well here.

Instant Gratification for Ontology Development While the benefits of existing alignment tests were mostly cropped by the developers of tools, the new collection will aim at remunerating the developers of ontologies themselves. The collection will be equipped with a simple HTML-based front end giving access not only to the ontologies themselves (via a conventional query interface) but also to directly usable alignment and distributed reasoning results. Initial version of the HTML front end is described in [Svab *et al.*, 2005]. As the ontologies in the collection mirror the structure of real-world entities (namely, actual conference series), their alignment can give insights into this structure. Some candidate pay-off tasks, stimulating the development of further additions, are:

– Advertising the conference to the right target group, namely, offering the potential paper authors a conference that is thematically close, within reasonable time, in a certain (type of) location, with PC members from their institute/country and the like.
– Making the organizers aware of relationships (e.g., overlaps, personal links) with other conferences.
– Offering the conference organizers a suitable software tool that could support the organization of the event.

28.3 Thesaurus Alignment

The OAEI held in 2005 contained two different real world alignment tasks. One focussed on mapping Medical heavy-weight ontologies like OpenGalen and the other on mapping directory structures like Looksmart. The most widely used ontologies however are thesauri that lie in between these two kinds of ontologies in both richness and size. Thesauri are linguistic models that have been engineered to facilitate finding the right word to denote something. In libraries thesauri terms have been used to categorize publications ever since sorting books became necessary. Recently, thesauri have taken a leap in popularity, because the advent of the World Wide Web has made it easy for organizations to open their knowledge organization systems to the rest of the world. The Semantic Web community realized that semantic negotiation of such opened resources is necessary, which lead to the creation of the Simple Knowledge Organization System (SKOS[3].

SKOS SKOS consists of thee vocabularies: The SKOS Core Vocabulary, which contains the main classes and properties that are necessary to encode everything from controlled vocabularies to thesauri; the SKOS Mapping Vocabulary, which contains properties to create mappings between SKOS vocabularies; and SKOS Extensions, which contains domain-specific extensions to the SKOS Core and SKOS Mapping Vocabularies.

SKOS Collections Many organizations world-wide have started converting their thesauri to SKOS. Two such organizations are the Food and Agriculture Organization (FAO) of the United Nations and the United States National Agricultural Library (NAL). The FAO has converted their multilingual AGROVOC thesaurus[4], which consists of more than 16.000 terms into SKOS and is currently working on extending it with OWL statements. AGROVOC covers many subjects related to food and agriculture, such as fishing, famine and forestry. The NAL will release a SKOS version of their (monolingual english) NAL Agricultural Thesaurus in january 2006 [5]. The NAL

[3]`http://www.w3.org/2004/02/skos`, see Deliverable D2.2.6 [Euzenat *et al.*, 2005b].
[4]`http://www.fao.org/agrovoc`
[5]`http://agclass.nal.usda.gov/agt`

thesaurus will consist of more than 41.000 terms covering an equally broad spectrum of subjects like food, agriculture, and the environment. Both thesauri are used to index large actively maintained research libraries, which are heavily used by researchers all over the world, such as food product developers, researchers investigating food-safety, and environmental policy makers.

Thesaurus Mapping Task Proposal A possible additional track for a further OAEI campaign could focus on creating a SKOS mapping between the SKOS versions of the AGROVOC and NAL thesauri. The mapping task is suitable for the OAEI, because of the following reasons:

– Both thesauri are large.
– They are widely used.
– The thesauri cover much of the same subjects.
– The concepts covered in the thesauri are understandable to people that are not a domain expert (For example, semantic web researchers.)
– The SKOS Mapping Vocabulary is an important, applied (would be) standard.
– Many of the corpora indexed with terms from the thesauri (instance data) are freely accessible on the web.

It however suffers from the following shortcommings:

– It is useful for thesauri aligners rather than ontology aligner.
– Delivering in SKOS is not prone to be integrated into the current do2002a platform (while delivering in the alignment API, will before the next do2002a provide SKOS generation).
– As for the anatomy example, there is currently no accepted mapping between these ontologies so the do2002a problem remain the same as this year.

28.4 Full real-world problem solving

One observation that was made is that we have trouble evaluating “real world” test cases. Moreover, by evaluating features of alignment, we do not evaluate their value in context, i.e., for solving real problems. In context, it can be possible to compare the performance of the systems without knowing an absolutely true correct alignment. We could measure if the system performs better as a whole in a (semi-)operational context.

For that purpose, we would like to have proposal challenge from real users who need ontology alignment. The ontology would be provided by the use case provider as well as the success criterion and some infrastructure for pluging alignments to be used. The organisers could provide help for setting the do2002a protocol.

This would help us having an independently submitted and independently evaluated real-world problem to solve; this would help the submitter having help from the community as a whole to solve her problem. Moreover, it is more gratifying for participant to know that they have contributed improving the solution to some real-world problem.

Part V

Conclusions

The work reported in this deliverable shows that KnowledgeWeb has succeeded in setting up an alignment challenge that attracts attention not only inside the network, which is demonstrated by the participation of research institutes from the North America and Asia in the two campaigns so far. The alignment challenge follows a clear methodology that has been described in detail in deliverable 2.2.3 and refined in this deliverable and has lead to advances in the state of the art in ontology alignment techniques which is demonstrated in part 1 of this deliverable.

A major aspect of this deliverable is to show how both the design of the do2002a methodology as well as the methods competing in the challenge evolve. The tests that have been run this year are harder and more complete than those of last year. However, more teams participated and the results tend to be better. This shows that, as expected, the field of ontology alignment is getting stronger (and we hope that do2002a is contributing to this progress). Reading the papers of the participants should help people involved in ontology matching to find what make these algorithms work and what could be improved.

Another noteworthy feature is that most of the tools that have been presented here are available on the web and many of them open source. This contributes independent scrutiny and improvement of the field as a whole.

The Ontology Alignment Evaluation Initiative has been created to carry this do2002a effort outside Knowledge web. It will continue these tests by improving both test cases and test methodology for being more accurate. It can be found at:

`http://oaei.inrialpes.fr`.

A number of obstacles to a successful do2002a of ontology alignment technology have been identified in the context of the do2002a. For example, problems related to quality measures for alignments are discussed in section 10 and proposals for measures that are better suited for the purpose than standard measures from information retrieval are made.

The most severe obstacle to a successful do2002a that has been identified is the lack of suitable test data. The reason for this is that there is a conflict between the need to measure the quality of the generated alignments and the wish to have realistic alignment problems. At the current stage, the data sets that support do2002a in terms of a standard alignment automatically generated ones can be compared are either rather small like the Benchmark data set in the challenge or rather inexpressive like the Directory data set in the challenge. Other data sets that are both, large and complex like the Anatomy data set in this years challenge are very hard to evaluate as it is entirely unclear how a correct mapping looks like. Even the Directory data set could only be evaluated for completeness, but not for correctness. We try to address this problem by suggesting a number of alternative data sets for future challenges, but the problem described above seems to be a fundamental dilemma.

A way to overcome this dilemma is to judge alignments not in terms of precision and recall but in terms of the usefulness of the generated mappings with respect to a concrete application.

The future plans for the Ontology Alignement Evaluation Initiative and Knowledge web work package 2.2 are certainly to go ahead and improving the functioning of these do2002a campaign. This most surely involves:

- Finding new real world cases;
- Improving the tests along the lesson learned;
- Accepting continuous submissions (through validation of the results);
- Improving the measures to go beyond precision and recall.

Chapter 29

OAEI-2005: organization

The increasing number of methods available for schema matching/ontology integration suggests the need to establish a consensus for do2002a of these methods. The Ontology Alignment Evaluation Initiative[1] is now a coordinated international initiative that has been set up for organizing do2002a of ontology matching algorithms. After the two events organized in 2004 (namely, the Information Interpretation and Integration Conference (I3CON) and the EON Ontology Alignment Contest [Sure *et al.*, 2004]), this year one unique do2002a campaign is organized. Its outcome is presented at the Workshop on Integrating Ontologies held in conjunction with K-CAP 2005 at Banff (Canada) on October 2, 2005. Since last year, we have set up a web site, improved the software on which the tests can be evaluated and set up some precise guidelines for running these tests. We have taken into account last year's remarks by (1) adding more coverage to the benchmark suite and (2) elaborating two real world test cases (as well as addressing other technical comments).

This chapter serves as a presentation to the 2005 do2002a campaign and introduction to the results provided by the some of the systems presented in the previous papers.

29.1 Goals

Last year events demonstrated that it is possible to evaluate ontology alignment tools. One intermediate goal of this year is to take into account the comments from last year contests. In particular, we aimed at improving the tests by widening their scope and variety. Benchmark tests are more complete (and harder) than before. Newly introduced tracks are more 'real-world' and of a considerable size. The main goal of the Ontology Alignment Evaluation is to be able to compare systems and algorithms on the same basis and to allow drawing conclusions about the best strategies. Our ambition is that from such challenges, the tool developers can learn and improve their systems.

29.2 General methodology

We present below the general methodology for the 2005 campaign. In this we took into account many of the comments made during the previous campaign.

[1]`http://oaei.inrialpes.fr`

29.2.1 Alignment problems

This year's campaign consists of three parts: it features two real world blind tests (anatomy and directory) in addition to the systematic benchmark test suite. By blind tests it is meant that the result expected from the test is not known in advance by the participants. The do2002a organizers provide the participants with the pairs of ontologies to align as well as (in the case of the systematic benchmark suite only) expected results. The ontologies are described in OWL-DL and serialized in the RDF/XML format. The expected alignments are provided in a standard format expressed in RDF/XML [Euzenat, 2004].

– Like for last year's EON contest, a systematic benchmark series has been produced. The goal of this benchmark series is to identify the areas in which each alignment algorithm is strong and weak. The test is based on one particular ontology dedicated to the very narrow domain of bibliography and a number of alternative ontologies of the same domain for which alignments are provided.

– The directory real world case consists of aligning web sites directory (like open directory or Yahoo's). It is more than two thousand elementary tests.

– The anatomy real world case covers the domain of body anatomy and consists of two ontologies with an approximate size of several 10k classes and several dozen of relations.

The do2002a has been processed in three successive steps.

29.2.2 Preparatory phase

The ontologies and alignments of the do2002a have been provided in advance during the period between June 1st and July 1st. This was the occasion for potential participants to send observations, bug corrections, remarks and other test cases to the organizers. The goal of this primary period is to be sure that the delivered tests make sense to the participants. The feedback is important, so all participants were strongly invited to provide it. The final test base has been released on July 4th. The tests did only change after this period for ensuring a better and easier participation.

29.2.3 Execution phase

During the execution phase the participants have used their algorithms to automatically match the ontologies of both part. The participants were required to only use one algorithm and the same set of parameters for all tests. Of course, it is regular to select the set of parameters that provide the best results. Beside the parameters the input of the algorithms must be the two provided ontology to align and any general purpose resource available to everyone (that is no resource especially designed for the test). In particular, the participants should not use the data (ontologies and results) from other test sets to help their algorithm.

The participants have provided their alignment for each test in the Alignment format and a paper describing their results. In an attempt to validate independently the results, they were required to provide a link to their program and parameter set used for obtaining the results.

Name	System	Benchmarks	Directory	Anatomy	Validated	Relations	Confidence
U. Karlsruhe	FOAM	$\surd$	$\surd$			=	cont
U. Montréal/INRIA	OLA	$\surd$	$\surd$		$\surd$	=	cont
IRST Trento	CtxMatch 2	$\surd$	$\surd$			=, $\leq$	1/0
U. Southampton	CMS	$\surd$	$\surd$	$\surd$		=	1/0
Southeast U. Nanjin	Falcon	$\surd$	$\surd$	$\surd$	$\surd$	=	1/0
UC. Dublin	?	$\surd$	$\surd$			=	cont
CNR/Pisa	OMAP	$\surd$	$\surd$			=	1/0

Table 29.1: Participants and the state of the state of their submissions. Confidence is given as 1/0 or continuous values.

29.2.4 Evaluation phase

The organizers have evaluated the results of the algorithms used by the participants and provided comparisons on the basis of the provided alignments. In the case of the real world ontologies only the organizers did the do2002a with regard to the withheld alignments. The standard do2002a measures are precision and recall computed against the reference alignments. For the matter of aggregation of the measures we have computed a true global precision and recall (not a mere average). We have also computed precision/recall graphs for some of the participants (see below). Finally, in an experimental way, we have attempted this year at reproducing the results provided by participants (validation).

29.3 Comments on the execution

We had more participants than last year's event and it is easier to run these tests (qualitatively we had less comments and the results were easier to analyze). We summarize the list of participants in Table 29.1. As can be seen, not all participants provided results for all the tests and not all system were correctly validated. However, when the tests are straightforward to process (benchmarks and directory), participants provided results. The main problems with the anatomy test was its size. We also mentioned the kind of results sent by each participant (relations and confidence).

We note that the time devoted for performing these tests (three months) and the period allocated for that (summer) is relatively short and does not really allow the participants to analyze their results and improve their algorithms. On the one hand, this prevents having algorithms really tuned for the test set, on the other hand, this can be frustrating for the participants. We should try to allow more time for participating next time.

Complete results are provided on http://oaei.inrialpes.fr/2005/results/. These are the only official results (the results presented here are only partial and prone to correction). The summary of results track by track is provided below.

Chapter 30

OAEI-2005: results

30.1 Benchmark

The benchmark test case improved on last year's base by providing new variations of the reference ontology (last year the test contained 19 individual tests while this year it contains 53 tests). These new tests are supposed to be more difficult. The other improvement was the introduction of other do2002a metrics (real global precision and recall as well as the generation of precision-recall graphs).

30.1.1 Test set

The systematic benchmark test set is built around one reference ontology and many variations of it. The participants have to match this reference ontology with the variations. These variations are focussing the characterization of the behavior of the tools rather than having them compete on real-life problems. The ontologies are described in OWL-DL and serialized in the RDF/XML format. Since the goal of these tests is to offer some kind of permanent benchmarks to be used by many, the test is an extension of last year EON Ontology Alignment Contest. Test numbering (almost) fully preserves the numbering of the first EON contest.

The reference ontology is based on the one of the first EON Ontology Alignment Contest. It is improved by comprising a number of circular relations that were missing from the first test. The domain of this first test is Bibliographic references. It is, of course, based on a subjective view of what must be a bibliographic ontology. There can be many different classifications of publications (based on area, quality, etc.). We choose the one common among scholars based on mean of publications; as many ontologies below (tests #301-304), it is reminiscent to BibTeX. The reference ontology is that of test #101. It contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. The reference ontology is put in the context of the semantic web by using other external resources for expressing non bibliographic information. It takes advantage of FOAF[1] and iCalendar[2] for expressing the People, Organization and Event concepts. Here are the external reference used:

[1]`http://xmlns.com/foaf/0.1/`

[2]`http://www.w3.org/2002/12/cal/`

algo	edna		falcon		foam		ctxMatch2-1		dublin20		cms		omap		ola	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	0.96	1.00	1.00	1.00	0.98	0.65	0.10	0.34	1.00	0.99	0.74	0.20	0.96	1.00	1.00	1.00
2xx	0.41	0.56	0.90	0.89	0.89	0.69	0.08	0.23	0.94	0.71	0.81	0.18	0.31	0.68	0.80	0.73
3xx	0.47	0.82	0.93	0.83	0.92	0.69	0.08	0.22	0.67	0.60	0.93	0.18	0.93	0.65	0.50	0.48
H-means	0.45	0.61	0.91	0.89	0.90	0.69	0.08	0.24	0.92	0.72	0.81	0.18	0.35	0.70	0.80	0.74

Table 30.1: Means of results obtained by participants (corresponding to harmonic means)

– http://www.w3.org/2002/12/cal/#:Vevent (defined in http://www.w3.org/2002/12/cal/ical.n3 and supposedly in http://www.w3.org/2002/12/cal/ical.rdf)
– http://xmlns.com/foaf/0.1/#:Person (defined in http://xmlns.com/foaf/0.1/index.rdf)
– http://xmlns.com/foaf/0.1/#:Organization (defined in http://xmlns.com/foaf/0.1/index.rdf)

This reference ontology is a bit limited in the sense that it does not contain attachment to several classes. Similarly the kind of proposed alignments is still limited: they only match named classes and properties, they mostly use the "=" relation with confidence of 1. There are still three group of tests in this benchmark:

– simple tests (1xx) such as comparing the reference ontology with itself, with another irrelevant ontology (the wine ontology used in the OWL primer) or the same ontology in its restriction to OWL-Lite;
– systematic tests (2xx) that were obtained by discarding some features of the reference ontology. The considered features were (names, comments, hierarchy, instances, relations, restrictions, etc.). The tests are systematically generated to as to start from some reference ontology and discarding a number of information in order to evaluate how the algorithm behave when this information is lacking. These tests were largely improved from last year by combining all feature discarding.
– four real-life ontologies of bibliographic references (3xx) that were found on the web and left mostly untouched (they were added xmlns and xml:base attributes).

Table 30.4 summarize what has been retracted from the reference ontology in the systematic tests. There are here 6 categories of alteration:

Name Name of entities that can be replaced by (R/N) random strings, (S)ynonyms, (N)ame with different conventions, (F) strings in another language than english.
Comments Comments can be (N) suppressed or (F) translated in another language.
Specialization Hierarchy can be (N) suppressed, (E)xpansed or (F)lattened.
Instances can be (N) suppressed
Properties can be (N) suppressed or (R) having the restrictions on classes discarded.
Classes can be (E)xpanded, i.e., replaced by several classes or (F)latened.

30.1.2 Results

Table 30.1 provide the consolidated results, by groups of tests. Table 30.5 contain the full results.

We display the results of participants as well as those given by some very simple edit distance algorithm on labels (edna). The computed values here are real precision and recall and not a simple average of precision and recall. This is more accurate than what has been computed last

algo	karlsruhe2		umontreal		fujitsu		stanford	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	NaN	0.00	0.57	0.93	0.99	1.00	0.99	1.00
2xx	0.60	0.46	0.54	0.87	0.93	0.84	0.98	0.72
3xx	0.90	0.59	0.36	0.57	0.60	0.72	0.93	0.74
H-means	0.65	0.40	0.52	0.83	0.88	0.85	0.98	0.77

Table 30.2: EON 2004 results with this year's aggregation method.

algo	edna		falcon		foam		ctxMatch2-1		dublin20		cms		omap		ola	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	0.96	1.00	1.00	1.00	0.98	0.65	0.10	0.34	1.00	0.99	0.74	0.20	0.96	1.00	1.00	1.00
2xx	0.66	0.72	0.98	0.97	0.87	0.73	0.09	0.25	0.98	0.92	0.91	0.20	0.89	0.79	0.89	0.86
3xx	0.47	0.82	0.93	0.83	0.92	0.69	0.08	0.22	0.67	0.60	0.93	0.18	0.93	0.65	0.50	0.48
H-means	0.66	0.78	0.97	0.96	0.74	0.59	0.09	0.26	0.94	0.88	0.65	0.18	0.90	0.81	0.85	0.83

Table 30.3: This year's results on EON 2004 test bench.

year.

As can be seen, the 1xx tests are relatively easy for most of the participants. The 2xx tests are more difficult in general while 3xx tests are not significantly more difficult than 2xx for most participants. The real interesting results is that there are significant differences across algorithms within the 2xx test series. Most of the best algorithms were combining different ways of finding the correspondence. Each of them is able to perform quite well on some tests with some methods. So the key issue seems to have been the combination of different methods (as described by the papers).

One algorithm, Falcon, seems largely dominant. But a group of other algorithms (Dublin, OLA, FOAM) are competing against each other, while the CMS and CtxMatch currently perform at a lower rate. Concerning these algorithm, CMS seems to privilege precision and performs correctly in this (OLA seems to have privileged recall with regard to last year). CtxMatch has the difficulty of delivering many subsumption assertions. These assertions are taken by our do2002a procedure positively (even if equivalence assertions were required), but since there are many more assertions than in the reference alignments, this brings the result down.

These results can be compared with last year's results given in Table 30.2 (with aggregated measures computed at new with the methods of this year). For the sake of comparison, the results of this year on the same test set as last year are given in Table 30.3. As can be expected, the two participants of both challenges (Karlsruhe2 corresponding to foam and Montréal/INRIA corresponding to ola) have largely improved their results. The results of the best participants this year are over or similar to those of last year. This is remarkable, because participants did not tune their algorithms to the challenge of last year but to that of this year (more difficult since it contains more test of a more difficult nature and because of the addition of cycles in them).

So, it seem that the field is globally progressing.

Because of the precision/recall trade-off, as noted last year, it is difficult to compare the middle group of systems. In order to assess this, we attempted to draw precision recall graphs. We provide in Figure 30.1 the averaged precision and recall graphs of this year. They involve only the results of all participants. However, the results corresponding to participants who provided

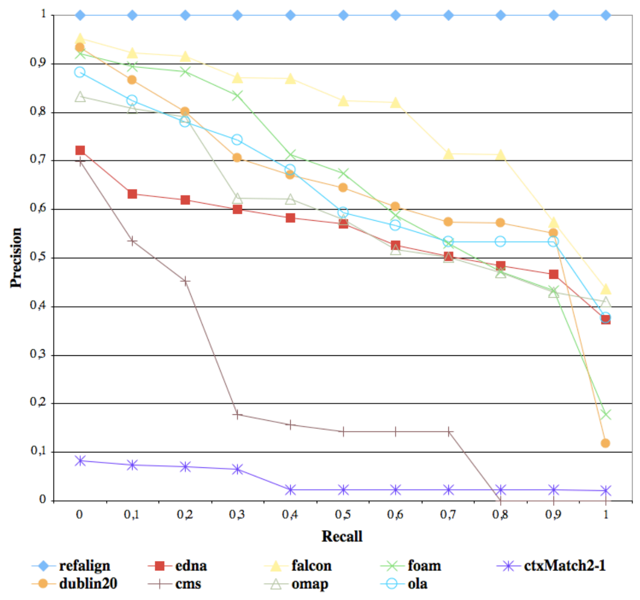


Figure 30.1: Precision-recall graphs

confidence measures equal to 1 or 0 (see Table 29.1) can be considered as approximation. Moreover, for reason of time these graphs have been computed by averaging the graphs of each tests (instead to pure precision and recall).

These graphs are not totally faithful to the algorithms because participants have cut their results (in order to get high overall precision and recall). However, they provide a rough idea about the way participants are fighting against each others in the precision recall space. It would be very useful that next year we ask for results with continuous ranking for drawing these kind of graphs.

30.1.3 Comments

As general comments, we remarks that it is still difficult for participants to provide results that correspond to the challenge (incorrect format, alignment with external entities). Because time is short and we try to avoid modifying provided results, this test is still a test of both algorithms and their ability to deliver a required format. However, some teams are really effective in this (and the same teams generally have their tools validated relatively easily).

The do2002a of algorithms like ctxMatch which provide many subsumption assertions is relatively inadequate. Even if the test can remain a test of inference equivalence. It would be useful to be able to count adequately, i.e., not negatively for precision, true assertions like owl:Thing subsuming another concept. We must develop new do2002a methods taken into account these assertions and the semantics of the OWL language.

30.2 Directory

30.2.1 Data set

The data set exploited in the web directories matching task was constructed from Google, Yahoo and Looksmart web directories as described in [Avesani *et al.*, 2005]. The key idea of the data set construction methodology was to significantly reduce the search space for human annotators. Instead of considering the full mapping task which is very big (Google and Yahoo directories have up to $3 * 10^5$ nodes each: this means that the human annotators need to consider up to $(3 * 10^5)^2 = 9 * 10^{10}$ mappings), it uses semi automatic pruning techniques in order to significantly reduce the search space. For example, for the dataset described in [Avesani *et al.*, 2005] human annotators consider only 2265 mappings instead of the full mapping problem.

The major limitation of the current dataset version is the fact that if it contains true positive mappings (i.e., it is correct), it does not contain them all (it is not complete). Notice that manually constructed mapping sets (such as ones presented for systematic tests) assume all the mappings except true positives to be true negatives (i.e., they are supposed to be complete). This limitation allows to use the dataset only for do2002a of Recall (since Recall is defined as ratio of correct mappings found by the system to the total number of correct mappings, this ratio is still meaningful if we only know a part of the correct mappings). At the same time measuring Precision necessarily requires the completeness in the dataset since Precision is defined as a ratio of correct mappings found by the system to all the mappings found by the system: in this case if we only know one part of the correct mapping it is possible that a better performing system have a worse precision on the test set.

The absence of completeness has significant implications on the testing methodology in general. In fact most of the state of the art matching systems can be tuned either to produce the results with better Recall or to produce the results with better Precision. For example, the system which produce the complete product relation on any input will always have 100% Recall. Therefore, the main methodological goal in the do2002a was to prevent Recall tuned systems from getting of unrealistically good results on the dataset. In order to accomplish this goal the double validation of the results was performed. The participants were asked for the binaries of their systems and were required to use the same sets of parameters in both web directory and systematic matching tasks. Then the results were double checked by organizers to ensure that the latter requirement is fulfilled by the authors. The process allow to recognize Recall tuned systems by analysis of systematic tests results.

The dataset originally was presented in its own format. The mappings were presented as pairwise relationships between the nodes of the web directories identified by their paths to root. Since the systems participating in the do2002a all take OWL ontologies as input the conversion of the dataset to OWL was performed. In the conversion process the nodes of the web directories were modeled as classes and classification relation connecting the nodes was modeled as rdfs:subClassOf relation. Moreover, in order to avoid presenting a too big challenge for matchers, the matching task was presented as 2265 tasks of finding the semantic relation holding between pathes to root in the web directories modeled as sub class hierarchies.

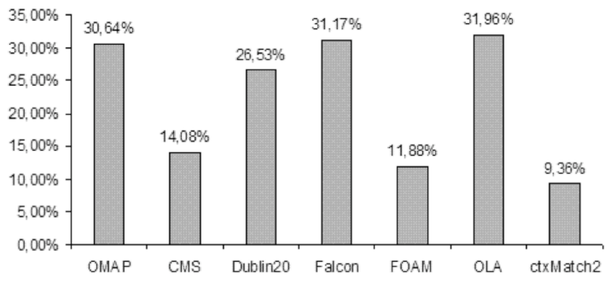


Figure 30.2: Recall for web directories matching task

30.2.2 Results

The results for web directory matching task are presented on Figure 30.2. As from the figure the web directories matching task is a very hard one. In fact the best systems found about 30% of mappings form the dataset (i.e., have Recall about 30%).

The do2002a results can be considered from two perspectives. On the one hand, they are good indicator of real world ontologies matching complexity. On the other hand, the results can provide information about the quality of the dataset used in the do2002a. The desired mapping dataset quality properties were defined in [Avesani *et al.*, 2005] as *Complexity*, *Discrimination capability*, *Incrementality* and *Correctness*. The first means that the dataset is "hard" for state of the art matching systems, the second that it discriminates among the various matching solutions, the third that it is effective in recognizing weaknesses in the state of the art matching systems and the fourth that it can be considered as a correct one.

The results of the do2002a give us some evidence for *Complexity* and *Discrimination capability* properties. As from Figure 30.2 TaxME dataset is hard for state of the art matching techniques since there are no systems having Recall more than 35% on the dataset. At the same time all the matching systems together found about 60% of mappings. This means that there is a big space for improvements for state of the art matching solutions.

Consider Figure 30.3. It contains partitioning of the mappings found by the matching systems. As from the figure 44% of the mappings found by any of the matching systems was found by only one system. This is a good argument to the dataset *Discrimination capability* property.

30.2.3 Comments

The web directories matching task is an important step towards do2002a on the real world matching problems. At the same time there are a number of limitations which makes the task only an intermediate step. First of all the current version of the mapping dataset provides correct but not complete set of reference mappings. The new mapping dataset construction techniques can overcome this limitation. In the do2002a the mapping task was split to the the tiny subtasks. This strategy allowed to obtain results form all the matching systems participating in the do2002a. At

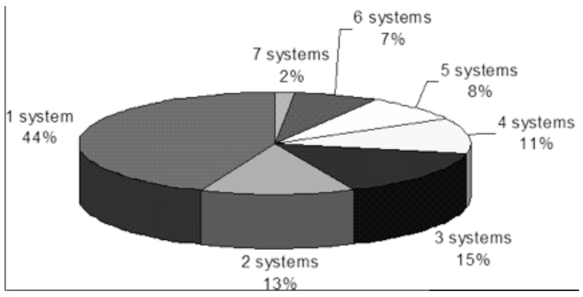


Figure 30.3: Partitioning of the mappings found by the matching systems

the same time it hides computational complexity of “real world” matching (the web directories have up to 10^5 nodes) and may affect the results of the tools relying on “look for similar siblings” heuristic.

The results obtained on the web directories matching task coincide well with previously reported results on the same dataset. According to [Avesani *et al.*, 2005] generic matching systems (or the systems intended to match any graph-like structures) have Recall from 30% to 60% on the dataset. At the same time the real world matching tasks are very hard for state of the art matching systems and there is a huge space for improvements in the ontology matching techniques.

30.3 Anatomy

30.3.1 Test set

The focus of this task is to confront existing alignment technology with real world ontologies. Our aim is to get a better impression of where we stand with respect to really hard challenges that normally require an enormous manual effort and requires in-depth knowledge of the domain. The task is placed in the medical domain as this is the domain where we find large, carefully designed ontologies. The specific characteristics of the ontologies are:

– Very large models: OWL models of more than 50MB !
– Extensive Class Hierarchies: ten thousands of classes organized according to different views on the domain.
– Complex Relationships: Classes are connected by a number of different relations.
– Stable Terminology: The basic terminology is rather stable and should not differ too much in the different model
– Clear Modeling Principles: The modeling principles are well defined and documented in publications about the ontologies

This implies that the task will be challenging from a technological point of view, but there is guidance for tuning matching approach that needs to be taken into account.

The ontologies to be aligned are different representations of human anatomy developed independently by teams of medical experts. Both ontologies are available in OWL format and mostly contain classes and relations between them. The use of axioms is limited.

The Foundational Model of Anatomy

The Foundational Model of Anatomy is a medical ontology developed by the University of Washington. We extracted an OWL version of the ontology from a Protege database. The model contains the following information:

– Class hierarchy;
– Relations between classes;
– Free text documentation and definitions of classes;
– Synonyms and names in different languages.

The OpenGalen Anatomy Model

The second ontology is the Anatomy model developed in the OpenGalen Project by the University of Manchester. We created an OWL version of the ontology using the export functionality of Protege. The model contains the following information:

– Concept hierarchy;
– Relations between concepts.

The task is to find alignment between classes in the two ontologies. In order to find the alignment, any information in the two models can be used. In addition, it is allowed to use background knowledge, that has not specifically been created for the alignment tasks (i.e., no hand-made mappings between parts of the ontologies). Admissible background knowledge are other medical terminologies such as UMLS as well as medical dictionaries and document sets. Further, results must not be tuned manually, for instance, by removing obviously wrong mappings.

At the time of printing we are not able to provide results of do2002a on this test.

30.3.2 Comments

We had very few participants able to even produce the alignments between both ontologies. This is mainly due to their inability to load these ontologies with current OWL tools (caused either by the size of the ontologies or errors in the OWL).

30.4 Result validation

As can be seen from the procedure, the results are not obtained independently. They have been computed from the alignment provided by the participants. In order to go one step further, we have attempted, this year, to generate the results obtained by the participants from their tools. The tools for which the results have been validated independently are marked in Table 29.1.

#	Name	Com	Hier	Inst	Prop	Class	Comment
101							Reference alignment
102							Irrelevant ontology
103							Language generalization
104							Language restriction
201	R						No names
202	R	N					No names, no comments
203		N					No comments (was missspelling)
204	C						Naming conventions
205	S						Synonyms
206	F	F					Translation
207	F						
208	C	N					
209	S	N					
210	F	N					
221			N				No specialisation
222			F				Flatenned hierarchy
223			E				Expanded hierarchy
224				N			No instance
225					R		No restrictions
226							No datatypes
227							Unit difference
228					N		No properties
229							Class vs instances
230						F	Flattened classes
231*						E	Expanded classes
232			N	N			
233			N		N		
236				N	N		
237			F	N			
238			E	N			
239			F		N		
240			E		N		
241			N	N	N		
246			F	N	N		
247			E	N	N		
248	N	N	N				
249	N	N		N			
250	N	N			N		
251	N	N	F				
252	N	N	E				
253	N	N	N	N			
254	N	N	N		N		
257	N	N		N	N		
258	N	N	F	N			
259	N	N	E	N			
260	N	N	F		N		
261	N	N	E		N		
262	N	N	N	N	N		
265	N	N	F	N	N		
266	N	N	E	N	N		
301							Real: BibTeX/MIT
302							Real: BibTeX/UMBC
303							Real: Karlsruhe
304							Real: INRIA

Table 30.4: Structure of the systematic benchmark test-case

algo	edna		falcon		foam		ctxMatch2-1		dublin20		cms		omap		ola	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
101	0.96	1.00	1.00	1.00	n/a	n/a	0.10	0.34	1.00	0.99	n/a	n/a	0.96	1.00	1.00	1.00
103	0.96	1.00	1.00	1.00	0.98	0.98	0.10	0.34	1.00	0.99	0.67	0.25	0.96	1.00	1.00	1.00
104	0.96	1.00	1.00	1.00	0.98	0.98	0.10	0.34	1.00	0.99	0.80	0.34	0.96	1.00	1.00	1.00
201	0.03	0.03	0.98	0.98	n/a	n/a	0.00	0.00	0.96	0.96	1.00	0.07	0.80	0.38	0.71	0.62
202	0.03	0.03	0.87	0.87	0.79	0.52	0.00	0.00	0.75	0.28	0.25	0.01	0.82	0.24	0.66	0.56
203	0.96	1.00	1.00	1.00	1.00	1.00	0.08	0.34	1.00	0.99	1.00	0.24	0.96	1.00	1.00	1.00
204	0.90	0.94	1.00	1.00	1.00	0.97	0.09	0.28	0.98	0.98	1.00	0.24	0.93	0.89	0.94	0.94
205	0.34	0.35	0.88	0.87	0.89	0.73	0.05	0.11	0.98	0.97	1.00	0.09	0.58	0.66	0.43	0.42
206	0.51	0.54	1.00	0.99	1.00	0.82	0.05	0.08	0.96	0.95	1.00	0.09	0.74	0.49	0.94	0.93
207	0.51	0.54	1.00	0.99	0.96	0.78	0.05	0.08	0.96	0.95	1.00	0.09	0.74	0.49	0.95	0.94
208	0.90	0.94	1.00	1.00	0.96	0.89	0.09	0.28	0.99	0.96	1.00	0.19	0.96	0.90	0.94	0.94
209	0.35	0.36	0.86	0.86	0.78	0.58	0.05	0.11	0.68	0.56	1.00	0.04	0.41	0.60	0.43	0.42
210	0.51	0.54	0.97	0.96	0.87	0.64	0.05	0.08	0.96	0.82	0.82	0.09	0.88	0.39	0.95	0.94
221	0.96	1.00	1.00	1.00	1.00	1.00	0.12	0.34	1.00	0.99	1.00	0.27	0.96	1.00	1.00	1.00
222	0.91	0.99	1.00	1.00	0.98	0.98	0.11	0.31	1.00	0.99	1.00	0.23	0.96	1.00	1.00	1.00
223	0.96	1.00	1.00	1.00	0.99	0.98	0.09	0.34	0.99	0.98	0.96	0.26	0.96	1.00	1.00	1.00
224	0.96	1.00	1.00	1.00	1.00	0.99	0.10	0.34	1.00	0.99	1.00	0.27	0.96	1.00	1.00	1.00
225	0.96	1.00	1.00	1.00	0.00	0.00	0.08	0.34	1.00	0.99	0.74	0.26	0.96	1.00	1.00	1.00
228	0.38	1.00	1.00	1.00	1.00	1.00	0.12	1.00	1.00	1.00	0.74	0.76	0.92	1.00	1.00	1.00
230	0.71	1.00	0.94	1.00	0.94	1.00	0.08	0.35	0.95	0.99	1.00	0.26	0.89	1.00	0.95	0.97
231	0.96	1.00	1.00	1.00	0.98	0.98	0.10	0.34	1.00	0.99	1.00	0.27	0.96	1.00	1.00	1.00
232	0.96	1.00	1.00	1.00	1.00	0.99	0.12	0.34	1.00	0.99	1.00	0.27	0.96	1.00	1.00	1.00
233	0.38	1.00	1.00	1.00	1.00	1.00	0.12	1.00	1.00	1.00	0.81	0.76	0.92	1.00	1.00	1.00
236	0.38	1.00	1.00	1.00	1.00	1.00	0.09	1.00	1.00	1.00	0.74	0.76	0.92	1.00	1.00	1.00
237	0.91	0.99	1.00	1.00	1.00	0.99	0.11	0.31	1.00	0.99	1.00	0.23	0.95	1.00	0.97	0.98
238	0.96	1.00	0.99	0.99	1.00	0.99	0.07	0.34	0.99	0.98	0.96	0.26	0.96	1.00	0.99	0.99
239	0.28	1.00	0.97	1.00	0.97	1.00	0.14	1.00	0.97	1.00	0.71	0.76	0.85	1.00	0.97	1.00
240	0.33	1.00	0.97	1.00	0.94	0.97	0.10	1.00	0.94	0.97	0.71	0.73	0.87	1.00	0.97	1.00
241	0.38	1.00	1.00	1.00	1.00	1.00	0.12	1.00	1.00	1.00	0.81	0.76	0.92	1.00	1.00	1.00
246	0.28	1.00	0.97	1.00	0.97	1.00	0.14	1.00	0.97	1.00	0.71	0.76	0.85	1.00	0.97	1.00
247	0.33	1.00	0.94	0.97	0.94	0.97	0.10	1.00	0.94	0.97	0.71	0.73	0.87	1.00	0.97	1.00
248	0.06	0.06	0.84	0.82	0.89	0.51	0.00	0.00	0.71	0.25	0.25	0.01	0.82	0.24	0.59	0.46
249	0.04	0.04	0.86	0.86	0.80	0.51	0.00	0.00	0.74	0.29	0.25	0.01	0.81	0.23	0.59	0.46
250	0.01	0.03	0.77	0.70	1.00	0.55	0.00	0.00	1.00	0.09	0.00	0.00	0.05	0.45	0.30	0.24
251	0.01	0.01	0.69	0.69	0.90	0.41	0.00	0.00	0.79	0.32	0.25	0.01	0.82	0.25	0.42	0.30
252	0.01	0.01	0.67	0.67	0.67	0.35	0.00	0.00	0.57	0.22	0.25	0.01	0.82	0.24	0.59	0.52
253	0.05	0.05	0.86	0.85	0.80	0.40	0.00	0.00	0.76	0.27	0.25	0.01	0.81	0.23	0.56	0.41
254	0.02	0.06	1.00	0.27	0.78	0.21	0.00	0.00	NaN	0.00	0.00	0.00	0.03	1.00	0.04	0.03
257	0.01	0.03	0.70	0.64	1.00	0.64	0.00	0.00	1.00	0.09	0.00	0.00	0.05	0.45	0.25	0.21
258	0.01	0.01	0.70	0.70	0.88	0.39	0.00	0.00	0.79	0.32	0.25	0.01	0.82	0.25	0.49	0.35
259	0.01	0.01	0.68	0.68	0.61	0.34	0.00	0.00	0.59	0.21	0.25	0.01	0.82	0.24	0.58	0.47
260	0.00	0.00	0.52	0.48	0.75	0.31	0.00	0.00	0.75	0.10	0.00	0.00	0.05	0.86	0.26	0.17
261	0.00	0.00	0.50	0.48	0.63	0.30	0.00	0.00	0.33	0.06	0.00	0.00	0.01	0.15	0.14	0.09
262	0.01	0.03	0.89	0.24	0.78	0.21	0.00	0.00	NaN	0.00	0.00	0.00	0.03	1.00	0.20	0.06
265	0.00	0.00	0.48	0.45	0.75	0.31	0.00	0.00	0.75	0.10	0.00	0.00	0.05	0.86	0.22	0.14
266	0.00	0.00	0.50	0.48	0.67	0.36	0.00	0.00	0.33	0.06	0.00	0.00	0.01	0.15	0.14	0.09
301	0.48	0.79	0.96	0.80	0.83	0.31	0.10	0.07	0.74	0.64	1.00	0.13	0.94	0.25	0.42	0.38
302	0.31	0.65	0.97	0.67	0.97	0.65	0.14	0.27	0.62	0.48	1.00	0.17	1.00	0.58	0.37	0.33
303	0.40	0.82	0.80	0.82	0.89	0.80	0.04	0.29	0.51	0.53	1.00	0.18	0.93	0.80	0.41	0.49
304	0.71	0.95	0.97	0.96	0.95	0.96	0.11	0.26	0.75	0.70	0.85	0.22	0.91	0.91	0.74	0.66
H-means	0.45	0.61	0.91	0.89	0.90	0.69	0.08	0.24	0.92	0.72	0.81	0.18	0.35	0.70	0.80	0.74

Table 30.5: Full results

Chapter 31

OAEI-2005: lesson learned and improvements

Beside the results of the do2002a properly speaking, there is a number of lessons that can be taken from running it. We consider below a number of them before providing some future plans linked to these remarks.

31.1 Lesson learned

From the 2005 OAEI campaign we can draw the following lessons:

More tools It seems that there are more and more tools able to jump in this kind of tests. This is a measure of the increase in interoperability of the tools developed for matching ontologies. This is also a call for carrying on these experiments (they are possible and people participate).

Tool robustness Contrary to last year it seems that the tools are more robusts and people deal with more wider implementation of OWL. However, this can be that we tuned the tests so that no one has problems. But our global impression is that both tools and the way people design OWL ontologies have improved.

Few suited corpus Contrary to what many people think, it is not that easy to find ontological corpora suitable for this do2002a test. From the proposals we had from last year, only one proved to be usable and with great difficulty (on size, conformance and juridical aspects). One could claim that matching thus solve no problem at all or that we do not yet have developed ontologies of significant size that people are ready to release.

Test realism The extension of the benchmark tests towards more coverage of the space is relatively systematic. However, it would be interesting and certainly more realistic, instead of crippling all names to do it for some random proportion of them (5% 10% 20% 40% 60% 100% random change). This has not been done for reason of time.

Size problems The real world benchmarks were huge benchmarks. Two different strategies have been taken with them: cutting them in a huge set of tiny benchmark or providing them as is. The first solution brings us away from “real world”, while the second one raised serious problems to the participants. It would certainly be worth designing these tests in order to assess the current limitation of the tools by providing an increasingly large sequence of such tests (0.1%, 1%, 10%, 100% of the corpus for instance).

Difficult validation Validation of the results is quite difficult to establish. Problems for evaluating the directory test have been mentioned as well as problems in evaluating the results of semantic matchers whose goal is correctness and completeness rather than precision and recall. These measures are related but not equivalent. For dealing with these problems which are typically semantic problems, measures that take semantic into account must be developed.

31.2 Future plans

In order to address these problems, several number actions can be taken and will be considered for future do2002as:

Real real world example This first measure has been suggested by one of the participant at the workshop. Indeed, the real world tests used this year can be criticised for not being totally natural: one of them split huge ontologies in pieces and the other one changed the ontology language. Moreover, their do2002a is difficult. One way to reduce this problem would be to ask someone with real problems, with a real interest to see ontology matching at work to submit the problem and to evaluate it (or to provide the criterion). This would have the advantage of some test case not made by researchers (so less suspect to bias) and solving a real problem. For that purpose, we proposed to find some interested party, preferably from the industry sector, with an ontology matching need, to provide ontologies and to evaluate the results in function of its problem. A call has been posted on the OAEI website.

New measures and do2002a techniques Since last year we made some progress in do2002a techniques (in particular with the computation of precision/recall graphs). However, the results are still not satisfying. Thus we are working on providing better do2002a measures and methodologies. A number of these have already been investigated in depth and are presented in the next part of this document.

Sampling tests It becomes clear that if we want to assess the scalability of the proposed methods, it would be very useful to propose versions of the tests of different size. In particular, this will be done with particularly large ontologies. It may also be useful to have some randomness in the systematically generated tests of the benchmark suite. So we will work toward this goal.

Chapter 32

Introduction to the OAEI-2006 campaign

This chapter serves as an introduction to the do2002a campaign of 2006 and to the results provided in the following chapters. We present the general methodology for the 2006 campaign as it was defined and report its execution. Specifically, first we introduce OAEI-2006 tracks and test cases (§32.1). Then, we discuss the preparatory, execution and do2002a phases of the campaign, respectively (§32.2, §32.3, and §32.4). Finally, we comment on the OAEI-2006 campaign execution (§32.5).

The results of the OAEI-2006 campaign have been published in [Shvaiko *et al.*, 2006].

32.1 Tracks and test cases

The OAEI-2006 campaign has consisted of four tracks gathering six data sets and different do2002a modalities.

The benchmark track: Like in previous campaigns, systematic benchmark series have been produced. The goal of this benchmark series is to identify the areas in which each matching algorithm is strong or weak. The test is based on one particular ontology dedicated to the very narrow domain of bibliography and a number of alternative ontologies of the same domain for which alignments are provided.

The expressive ontologies track:

- **Anatomy:** The anatomy real world case covers the domain of body anatomy and consists of two ontologies with an approximate size of several ten thousands classes and several dozens of relations.
- **Jobs:** The jobs test case is an industry evaluated real world business case. A company has a need to improve job portal functionality with semantic technologies. To enable higher precision in retrieval of relevant job offers or applicant profiles, OWL ontologies from the employment sector are used to describe jobs and job seekers and matching with regard to these ontologies provides the improved results. For confidentiality reasons, the test is run by the company team (WorldWideJobs - WWJ GmbH) with software provided by the participants.

It turned out that due to changes in management of the company (WorldWideJobs - WWJ GmbH) this test case has not been completed, and therefore, we do not provide its results.

The directories and thesauri track:

Directory: The directory real world case consists of matching web directories, such as open directory, Google and Yahoo. It has more than four thousands of elementary tests.

Food: Two SKOS thesauri about food have to be matched using relations from the SKOS mapping vocabulary. Samples of the results are evaluated by domain experts.

The conference track and consensus workshop: Participants have been asked to freely explore a collection of conference organization ontologies (the domain being well understandable for every researcher). This effort was expected to materialize in usual alignments as well as in interesting individual correspondences ("nuggets"), aggregated statistical observations and/or implicit design patterns. There is no a priori reference alignment. For a selected sample of correspondences, consensus was sought at the workshop and the process of reaching consensus was recorded.

Table 32.1 summarizes the variation in the results expected from these tests.

test	language	relations	confidence	modalities
benchmarks	OWL	=	[0 1]	open
anatomy	OWL	=	1	blind
jobs	OWL	=	[0 1]	external
directory	OWL	=	1	blind
food	SKOS	narrowMatch, exactMatch, broadMatch	1	blind+consensual
conference	OWL-DL	$=, \leq$	1	blind+consensual

Table 32.1: Characteristics of test cases (open do2002a is done with already published expected results, blind do2002a is done by organizers from reference alignments unknown to the participants, consensual do2002a is obtained by reaching consensus over the found results and external do2002a is performed independently of the organizers by running the actual systems).

32.2 Preparatory phase

The ontologies and alignments of the do2002a have been provided in advance during the period between June 1st and June 28th. This gave potential participants the occasion to send observations, bug reports, remarks and other test cases to the organizers. The goal of this preparatory period is to ensure that the delivered tests make sense to the participants. The tests still evolved after this period, but only for ensuring a better participation to the tests. The final test base was released on August 23rd.

32.3 Execution phase

During the execution phase the participants used their systems to automatically match the ontologies from the test cases. Participants have been asked to use one algorithm and the same set of parameters for all tests in all tracks. It is fair to select the set of parameters that provide the best results (for the tests where results are known). Beside parameters, the input of the algorithms must be the two ontologies to be matched and any general purpose resource available to everyone, i.e., no resource especially designed for the test. In particular, the participants should not use the data (ontologies and reference alignments) from other test sets to help their algorithms.

In most cases ontologies are described in OWL-DL and serialized in the RDF/XML format. The expected alignments are provided in the Alignment format expressed in RDF/XML. All the participants also provided the papers that are published in [Shvaiko *et al.*, 2006] and a link to their systems and their configuration parameters.

32.4 Evaluation phase

The organizers have evaluated the results of the algorithms used by the participants and provided comparisons on the basis of the provided alignments.

In order to ensure that it is possible to process automatically the provided results, the participants were requested to provide preliminary results by September 4th. In the case of blind tests only the organizers did the do2002a with regard to the withheld reference alignments. In the case of double blind tests, the participants provided a version of their system and the values of the parameters if any.

The standard do2002a measures are precision and recall computed against the reference alignments. For the matter of aggregation of the measures we used weighted harmonic means (weights being the size of the true positives). This clearly helps in case of empty alignments. Another technique that was used is the computation of precision/recall graphs so it was advised that participants provide their results with a weight to each correspondence they found.

New measures addressing some limitations of precision and recall have also been used for testing purposes. These were presented at the OM-2006 workshop[1] discussion in order for the participants to provide feedback on the opportunity to use them in further do2002a.

32.5 Comments on the execution

In OAEI-2006 we had more participants than in previous years: 4 in 2004, 7 in 2005 and 10 in 2006. We also noted the increase in tools compliance and robustness: they had less problems to carry the tests and we had less problems to evaluate the results.

We have had not enough time so far to validate the results which have been provided by the participants. In 2005, validating these results has proved feasible so we plan to do it again in future (at least for those participants who provided their systems).

We summarize the list of participants in Table 32.2. Similar to OAEI-2005 not all participants provided results for all tests. They usually did those which are easier to run, such as benchmark, directory and conference. The jobs line corresponds to the participants who have provided an

[1] `http://www.om2006.ontologymatching.org`

Test \ System	Falcon	HMatch	DSSim	COMA++	AUTOMS	JHU/APL	PRIOR	RiMOM	OCM	NIH	Total
Benchmark	√	√	√	√	√	√	√	√	√		9
Anatomy	√	√		√			√			√	5
Jobs	√	√		√	√		√		√		6
Directory	√	√		√	√		√	√	√		7
Food	√	√		√			√	√			5
Conference	√	√		√	√			√	√		6
Certified											
Confidence	√	√		√			√	√			5
Time					√		√	√	√	√	5

Table 32.2: Participants and the state of their submissions. Confidence is ticked when given as non boolean value. Time indicates when participants included execution time with their tests.

executable version of their systems. The variety of tests and the short time given to provide results have certainly prevented participants from considering more tests.

Like in 2005, the time devoted for performing these tests (three months) and the period allocated for that (summer) is relatively short and does not really allow the participants to analyze their results and improve their algorithms. On the one hand, this prevents having algorithms to be particularly tuned for the tests. On the other hand, this can be frustrating for the participants.

Chapter 33

The benchmark track

The goal of the benchmark tests is to provide a stable and detailed picture of each algorithm. For that purpose, the algorithms are run on systematically generated test cases. In particular, first we discuss the test cases (§33.1), and then we provide their results for the participated systems (§33.2).

33.1 Test set

The domain of this first test is Bibliographic references. It is, of course, based on a subjective view of what must be a bibliographic ontology. There can be many different classifications of publications, for example, based on area and quality. The one chosen here is common among scholars and is based on publication categories; as many ontologies (tests #301-304), it is reminiscent to BibTeX.

The systematic benchmark test set is built around one reference ontology and many variations of it. The reference ontology is that of test #101. The participants have to match this reference ontology with the variations. These variations are focusing the characterization of the behavior of the tools rather than having them compete on real-life problems. The ontologies are described in OWL-DL and serialized in the RDF/XML format. This reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals.

Since the goal of these tests is to offer some kind of permanent benchmarks to be used by many, the data set is an extension of the 2004 EON Ontology Alignment Contest. The reference ontology has been improved in 2005 by including circular relations that were missing from the first test. In 2006, we put the UTF-8 version of the tests as standard, the ISO-8859-1 being optional. Test numbering (almost) fully preserves the numbering of the first EON contest of 2004.

The kind of expected alignments is still limited: they only match named classes and properties, they mostly use the "=" relation with confidence of 1. There are three groups of tests in this benchmark:

Simple tests (1xx) such as comparing the reference ontology with itself, with another irrelevant ontology (the wine ontology used in the OWL primer) or the same ontology in its restriction to OWL-Lite;

Systematic tests (2xx) that were obtained by discarding features from some reference ontology. It aims at evaluating how an algorithm behaves when a particular type of information is lacking. The considered features were:

- *Name of entities* that can be replaced by random strings, synonyms, name with different conventions, strings in another language than English;
- *Comments* that can be suppressed or translated in another language;
- *Specialization hierarchy* that can be suppressed, expanded or flattened;
- *Instances* that can be suppressed;
- *Properties* that can be suppressed or having the restrictions on classes discarded;
- *Classes* that can be expanded, i.e., replaced by several classes or flattened.

Four real-life ontologies of bibliographic references (3xx) that were found on the web and left mostly untouched (there were added xmlns and xml:base attributes).

Full description of these tests can be found on the OAEI web site[1], see also Table 33.3 (p.251).

33.2 Results

Table 33.1 provides the consolidated results, by groups of tests. We display the results of participants as well as those given by some very simple edit distance algorithm on labels (edna). Like in 2005, the computed values are real precision and recall and not a simple average of precision and recall. The full results are on the OAEI web site.

These results show already that three systems are relatively close (COMA++, Falcon and RiMOM). The RiMOM system is slightly ahead of the others on these raw results. The DSSim system obviously favored precision over recall but its precision degrades with "real world" 3xx series. No system had strictly lower performance than edna.

The results have also been compared with the three measures proposed in [Ehrig *et al.*, 2005] (also reported in deliverable 2.2.4) in 2005 (symmetric, effort-based and oriented). These are generalization of precision and recall in order to better discriminate systems that slightly miss the target from those which are grossly wrong. The three measures provide the same results. This is not really surprising given the proximity of these measures. As expected, they can only improve over traditional precision and recall. The improvement affects all the algorithms, but this is not always strong enough for being reflected in the aggregated results. Moreover, the new measures do not dramatically change the do2002a of the participating systems.

Each algorithm has its best score with the 1xx test series. There is no particular order between the two other series. Again, it is more interesting to look at the 2xx series structure to distinguish the strengths of algorithms.

In 2006 the apparently best algorithms provided their results with confidence measures. It is thus possible to draw precision/recall curves in order to compare them. We provide in Figure 33.1 the precision and recall graphs of 2006. They involve only the results of participants who provided confidence measures different from 1 or 0 (see Table 32.2). They also feature the results for edit distances on class names (edna) and the results of Falcon in 2005 (denoted as Falcon-2005) (presentation of these results have also been provided with another presentation in deliverable 1.2.2.2.1). The graph for Falcon-2005 is not really accurate since it provided 1/0 alignments in 2005. This graph has been drawn with only technical adaptation of the technique used in TREC. Moreover, due to lack of time, these graphs have been computed by averaging the graphs of each of the tests (instead of pure precision and recall).

[1]`http://oaei.ontologymatching.org/2006/`

Algo	refalign		edna		AUTOMS		COMA++		DSSim		Falcon		HMatch		JHU/APL		OCM		PRIOR		RiMOM	
Test	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
1xx	1.00	1.00	0.96	1.00	0.94	1.00	1.00	1.00	1.00	0.98	1.00	1.00	0.91	1.00	1.00	1.00	0.95	1.00	1.00	1.00	1.00	1.00
2xx	1.00	1.00	0.90	0.49	0.94	0.64	0.96	0.82	0.99	0.49	0.91	0.85	0.83	0.51	0.20	0.86	0.93	0.51	0.95	0.58	0.97	0.87
3xx	1.00	1.00	0.94	0.61	0.91	0.70	0.84	0.69	0.90	0.78	0.89	0.78	0.78	0.57	0.18	0.50	0.89	0.51	0.85	0.80	0.83	0.82
Total	1.00	1.00	0.91	0.54	0.94	0.67	0.96	0.83	0.98	0.55	0.92	0.86	0.84	0.55	0.22	0.85	0.93	0.55	0.95	0.63	0.96	0.88
Symmetric	1.00	1.00	0.91	0.54	0.94	0.68	0.96	0.83	0.99	0.55	0.94	0.89	0.85	0.56	0.22	0.87	0.93	0.55	0.96	0.64	0.97	0.89
Effort-based	1.00	1.00	0.91	0.54	0.94	0.68	0.96	0.83	0.99	0.55	0.94	0.89	0.85	0.56	0.22	0.87	0.93	0.55	0.96	0.64	0.97	0.89
Oriented	1.00	1.00	0.91	0.54	0.94	0.68	0.96	0.83	0.99	0.55	0.94	0.89	0.85	0.56	0.22	0.87	0.93	0.55	0.96	0.64	0.97	0.89

Table 33.1: Means of results obtained by participants on the benchmark test case (corresponding to harmonic means).

Contrary to 2005, we have three systems competing at the highest level in 2006 (Falcon, COMA++ and RiMOM) and there is a gap between these and the other systems. We have compared the results of OAEI-2006 with the previous years on the basis of 2004 tests, see Table 33.2. The three best systems (Falcon, COMA++ and RiMOM) arrive at the level of 2005 best system (Falcon). However, no system outperforms it. Unfortunately no representant of the group of systems that followed Falcon in OAEI-2005 participated in OAEI-2006.

The best systems are at the level of 2005 best system (Falcon).

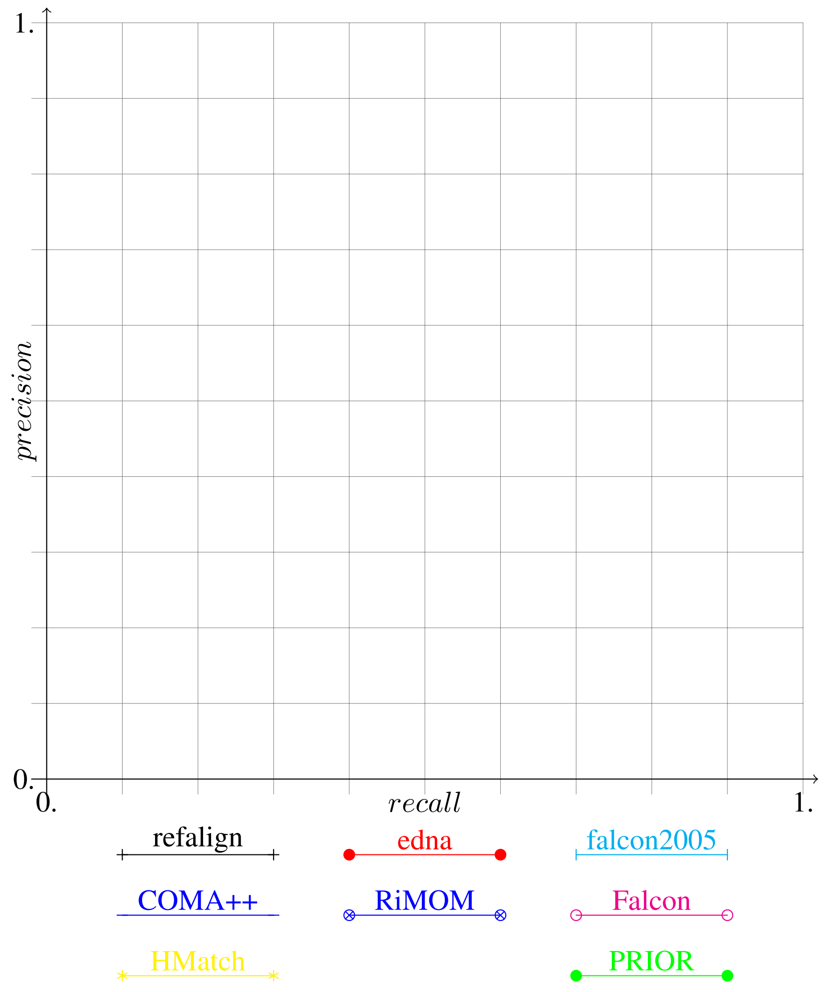


Figure 33.1: Precision/recall graphs for the systems which provided confidence values in their results.

Year	2004				2005		2006					
System	Fujitsu		Stanford		Falcon		RiMOM		Falcon		COMA++	
test	P	R	P	R	P	R	P	R	P	R	P	R
1xx	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2xx	0.93	0.84	0.98	0.72	0.98	0.97	1.00	0.98	0.97	0.97	0.99	0.97
3xx	0.60	0.72	0.93	0.74	0.93	0.83	0.83	0.82	0.89	0.78	0.84	0.69
H-means	0.88	0.85	0.98	0.77	0.97	0.96	0.97	0.96	0.97	0.95	0.98	0.94

Table 33.2: Evolution of the best scores over the years (on the basis of 2004 tests).

#	Name	Com	Hier	Inst	Prop	Class	Comment
101							Reference alignment
102							Irrelevant ontology
103							Language generalization
104							Language restriction
201	R						No names
202	R	N					No names, no comments
203		N					No comments (was missspelling)
204	C						Naming conventions
205	S						Synonyms
206	F	F					Translation
207	F						
208	C	N					
209	S	N					
210	F	N					
221			N				No specialisation
222			F				Flatenned hierarchy
223			E				Expanded hierarchy
224				N			No instance
225					R		No restrictions
226							No datatypes
227							Unit difference
228					N		No properties
229							Class vs instances
230						F	Flattened classes
231*						E	Expanded classes
232			N	N			
233			N		N		
236				N	N		
237			F	N			
238			E	N			
239			F		N		
240			E		N		
241			N	N	N		
246			F	N	N		
247			E	N	N		
248	N	N	N				
249	N	N		N			
250	N	N			N		
251	N	N	F				
252	N	N	E				
253	N	N	N	N			
254	N	N	N		N		
257	N	N		N	N		
258	N	N	F	N			
259	N	N	E	N			
260	N	N	F		N		
261	N	N	E		N		
262	N	N	N	N	N		
265	N	N	F	N	N		
266	N	N	E	N	N		
301							Real: BibTeX/MIT
302							Real: BibTeX/UMBC
303							Real: Karlsruhe
304							Real: INRIA

Table 33.3: Structure of the systematic benchmark test case.

Chapter 34

The expressive ontologies track

The focus of the anatomy test case used in the expressive ontology track is to confront existing matching technology with real world ontologies. Our aim here is to obtain a better impression of where the matching technology is positioned with respect to real challenges that normally require an enormous manual effort and an in-depth knowledge of the domain.

In particular, first we discuss the test case (§34.1). Then, we provide its early results (§34.2), and a followed up alignment cross-validation analysis for the participated systems (§34.3). Finally, we outline the major findings of this chapter (§34.4).

Some parts of the material presented in this chapter have been published in [Zhang and Bodenreider, 2007b].

34.1 Test set

The task is placed in the medical domain as this is the domain where we find large, carefully designed ontologies. The specific characteristics of the ontologies under consideration include:

- Very large models: OWL models of more than 50MBs.
- Extensive class hierarchies: ten thousands of classes organized according to different views on the domain.
- Complex relationships: classes are connected by a number of different relations.
- Stable terminology: the basic terminology is rather stable and should not differ substantially in the different models.
- Clear modeling principles: the modeling principles are well-defined and documented in publications about the ontologies.

As a consequence, the anatomy test case actually assesses existing matching systems with respect to two questions.

- Do existing approaches scale to very large models?
- Are existing approaches able to take advantage of well-documented modeling principles and knowledge about the domain?

The above two questions also mean that the goal of this test case is not to compare the performance of matching systems on a quantitative basis (though we provide some comparisons). We are rather interested in how many systems are actually able to create an alignment at all and in specific heuristics used for computing it.

The ontologies to be matched are different representations of human anatomy developed independently by teams of medical experts. Both ontologies are available in OWL format and mostly contain classes and relations between them. The use of axioms is limited.

34.1.1 The Foundational Model of Anatomy

The Foundational Model of Anatomy (FMA) has been developed by the University of Washington. It is an ontology describing the human anatomy including a taxonomy of body parts, information about anatomical structures and structure transformations. According to the developers the Foundational Model of Anatomy ontology contains approximately 75.000 classes and over 120.000 terms; over 2.1 million relationship instances from 168 relationship types link the FMA's classes into a coherent symbolic model.

We extracted an OWL version of the ontology from a Protégé database. The resulting model is in OWL-full as relations are defined between classes rather than instances.

34.1.2 Galen

The second ontology is the anatomy model developed in the OpenGalen Project by the University of Manchester. According to the creators, the ontology contains around 10.000 concepts covering somewhat more than standard textbook anatomy in terms of body parts, anatomical structures and relations between different parts and structures.

The ontology is freely available as a Protégé project file on the OpenGalen web page. We created an OWL version of the ontology using the export functionality of Protégé. The resulting ontology is in OWL-DL thus supporting logical reasoning about inconsistencies.

34.2 Initial results

The anatomy use case is part of the ontology alignment do2002a challenge for the second time now. While in 2005, none of the participants was in a position to submit a result for this data set. Almost all participants reported major difficulties in processing the ontologies due to their size and the fact that one of the models is in OWL full. At least these scalability problems appear somewhat to be solved this year. In the OAEI-2006 campaign, five out of ten participants submitted results for the anatomy data set. This clearly shows the advance of matching systems on the technical level and also suggests that matching technology is potentially ready for large scale applications.

On the content level the results are much harder to judge. Due to lack of a reference alignment, we were not able to provide a quantitative judgment and comparison of the different systems for the time of the OAEI-2006 results presentation at the Ontology Matching workshop[1]. Thus, we rather concentrated on the coverage of the ontologies, the degree of agreement amongst the matching systems and on the specific techniques the matching systems used in order to address this task.

A first observation is that none of the systems managed to reach a good coverage of the ontologies. Although both models contain several ten thousand concepts and the fact that we can assume

[1] `http://www.om2006.ontologymatching.org/`

a high degree of overlap in the two models, the systems were only able to produce correspondences for 2000 to 3000 concepts, which is less than 5% of the concepts of FMA.

We also found out that systems have severe difficulties with irregular concept names. The GALEN ontology contains a subset of concepts with highly irregular concept names. It turned out that only one system (COMA++) was able to determine correspondences for these concepts – at the expense of not being able to match any of the concept names with regular names.

A common pattern can be observed by looking at the actual methods used by the systems. Almost all systems use the linguistic similarity between class names and other features of the class description as a basis for determining match candidates. Normally, the systems combine different similarity measures. On top of this purely linguistic comparison, some systems also apply structural techniques. In particular, they translate the models into a graph structure and propagate the individual similarity in the graph structure. Only one of the systems (AOAS) actually used reasoning techniques to validate hypothesis and to determine matches based on the semantics of the models.

34.3 Cross-validation of the alignments

After the OAEI-2006 campaign, the results discussed so far have been further analyzed in [Zhang and Bodenreider, 2007b]. In particular, some of the results were reviewed based on cross-validation in order to obtain insights into the strengths and weaknesses of the various approaches. The key hypothesis of the cross-validation is that correspondences identified by several systems have a better chance to be valid.

34.3.1 Systems under consideration

The three matching systems analyzed in this study are the Anatomical Ontology Alignment System (AOAS), which participated in OAEI-2006 as the NIH system, PRIOR and Falcon. Two other systems participating in the OAEI-2006 campaign are not included in this review for the following reasons. Almost all correspondences identified by COMA++ were specific to this system and could not therefore contribute to cross-validation. The result files contributed by IsLab were not available when this study was performed. As most matching systems, the three systems under investigation rely on a combination of lexical and structural methods, based on the assumption that equivalent concepts across ontologies have similar names and similar relations to other concepts.

Notice that both PRIOR and Falcon allow partial matches between concept names (e.g., Adductor magnus of thigh matches Adductor magnus), while only minor term variations are allowed between matches by AOAS. Unlike AOAS or Falcon, PRIOR can exploit the anonymous concepts in GALEN. Finally, while AOAS only identifies correspondences between concepts, PRIOR and Falcon also find correspondences between relationships.

34.3.2 Methods and results

Overlap. First, the intersection among the lists of correspondences obtained by the three systems has been computed. This partitioned the set of all correspondences into subsets with respect to the systems that identified them (e.g., correspondences identified by AOAS and Falcon, but not by PRIOR). In particular, 1.429 matches were identified by all of the three systems, accounting for approximately 46%, 57% and 55% of concept matches in AOAS, Falcon and PRIOR, respectively.

The proportion of correspondences specific to one system varies largely, from 14% for Falcon to 39% for PRIOR, with 27% for AOAS.

Manual validation. Olivier Bodenreider manually reviewed for accuracy all correspondences not identified by AOAS. There are several reasons for explaining the bias towards this system. Unlike the other two systems, AOAS was developed specifically for matching anatomical ontologies. In a recent work [Zhang and Bodenreider, 2007a], those results were evaluated against a reference alignment established manually and against other systems. Recall was about .9 and most correspondences identified specifically by AOAS were deemed valid. Then, the correspondences were classified into the following categories: certain, possible (requires additional domain knowledge) and wrong. The objective of this cursory do2002a is primarily to quantify the false positives for Falcon and PRIOR and the false negatives for AOAS. As a result, 1.183 of the 1.383 (86%) correspondences not identified by AOAS were deemed invalid. More knowledge is required to establish the validity of half of the remaining 14%.

34.4 Discussion

For the first time since the anatomy data set has been used in the ontology alignment do2002a challenge, we are in a position, where we can actually compare the results of different matching systems. The results show that there is still a lot of work to do to make matching systems ready for real life applications. The problems above showed that differences in the naming scheme of classes can already cause matchers to fail on a significant subset of the vocabulary. It seems that existing matchers suffer from the need to balance precision and recall in determining correspondences. This conclusion is backed by results from other experiments, where it turned out that matching systems that produce highly precise correspondences miss many correspondences found by other systems. We conclude that using fixed thresholds to determine match candidates is not a good way for trading-off precision and recall.

Lexical matching constitutes an important step in ontology matching. Systems such as PRIOR focusing on bag-of-word matching rather than term matching miss many correspondences identified by the other two systems on the basis of exact matches of concept names. Compared to AOAS, Falcon uses a relaxed model of lexical similarity, based on edit distance. AOAS missed some correspondences due to improper segmentation of the original GALEN strings. For example, the string SupraHyoidMuscle was segmented at points where case changes, leading to the term supra hyoid muscle. However, the proper spelling for this term is suprahyoid muscle and the normalization algorithm used by AOAS could not match the two terms. In contrast, the relaxed approach to string matching employed by Falcon identified the two strings as a match. The analysis of the correspondences identified by Falcon and not AOAS revealed about 10 segmentation issues and 15 misspellings in GALEN (e.g., Mensicus for Meniscus). Conversely, the relaxed model of lexical resemblance can lead to “egregious” correspondences and therefore be extremely detrimental to the alignment. For example, Falcon identified a correspondence between Axillary artery (in the armpit) and Maxillary artery (near the mandible).

We were disappointed to see that only one system actually used some form of reasoning in order to take the meaning of the ontologies into account. As one of the major advantages of OWL is the ability to specify and reason about the semantics of concepts, it is at least surprising that this feature is not exploited by existing matchers. In fact, logical reasoning could be a way of becoming less dependent on the quality of certain similarity measures that obviously have some limitations when it comes to complex ontologies.

AOAS is the only system to fully take advantage of synonymy for the alignment. Some synonyms are provided by the FMA, but others come from the UMLS metathesaurus. In fact, it has been verified that most of the 856 correspondences identified by AOAS are indeed valid and involve such synonyms. This is the case, for example, of the correspondence between Aortic orifice and Ostium of aorta, and between Shoulder joint and Glenohumeral joint. The more conservative and linguistically-motivated approach to lexical similarity adopted by AOAS [McCray *et al.*, 1994] prevents a large number of false positives. However, it is also more sensitive to misspellings and segmentation issues, as well as missing synonyms. Overall, we believe that the benefit of preventing many false positives largely outweighs the few false negatives. It is clear why generic and domain-independent systems such as Falcon and PRIOR have adopted relaxed lexical models. The resources available for biomedicine (UMLS synonyms, domain-specific model of lexical resemblance) are not available for most domains. However, pairs of long terms encountered in anatomy often differing by one qualifier (e.g., for laterality) have an artificially high similarity value when compared with edit distance or in a vector space model. Calibrating the models for a particular domain is an issue that remains to be addressed.

In summary, the results of the anatomy test case have shown that there is some significant progress in terms of the maturity of matching technology. However, the results also show that there are still a lot of open problems with respect to producing good alignments on real life cases.

Chapter 35

The directories and thesauri track

This track includes two test cases: directory (§35.1) and food (§35.2).

35.1 The directory test case

The directory test case aims at providing a challenging task for ontology matchers in the domain of large web directories. The manual construction of the reference correspondences for the large applications is usually too demanding to the point of being infeasible, since the number of possible correspondences grows quadratically with the number of entities to be compared. This suggests the need for the development of semi-automatic approaches for acquiring the reference correspondences. First, we introduce the methodology used to build semi-automatically the datasets for evaluating recall (§35.1.1) and precision (§35.1.2). Then, we briefly summarize the web directories test set as used in OAEI-2006 (§35.1.3) and present its results for the participating systems (§35.1.4). Finally, we discuss the major findings of the web directories test case (§35.1.5).

35.1.1 A dataset for evaluating recall

We follow the semi-automatic method for an approximation of reference alignment proposed in [Avesani *et al.*, 2005] and applied it to the Google, Yahoo and Looksmart web directories. The key idea is to rely on a reference interpretation for entities (nodes), constructed by analyzing which documents have been classified in which nodes. The assumption is that the semantics of nodes can be derived from their pragmatics, namely from analyzing the documents that are classified under the given nodes. In particular, following on the work described in [Avesani *et al.*, 2005] we argue that the meaning of two nodes is equivalent if the sets of documents classified under those nodes have a meaningful overlap. The basic idea is therefore to compute the relationship hypotheses based on the co-occurence of documents.

Let us consider the example presented in Figure 35.1. Let N_1 be a node in the first taxonomy and N_2 be a node in the second taxonomy. D_1 and D_2 stand for the sets of documents classified under the nodes N_1 and N_2 respectively. A_2 denotes the documents classified in the ancestor node of N_2; C_1 denotes the documents classified in the children nodes of N_1. A simple *equivalence* measure is defined as follows:

(35.1)
$$Eq(N_1, N_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2| - |D_1 \cap D_2|}$$

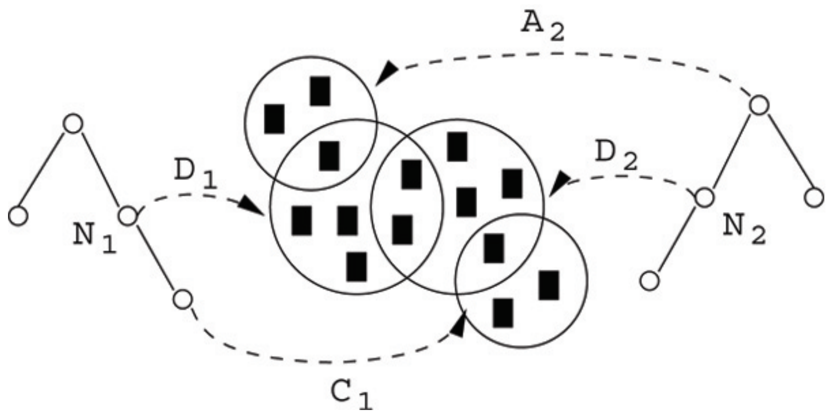


Figure 35.1: *TaxME*. Illustration of a document-driven similarity assessment.

Notice that the range of $Eq(N_1, N_2)$ is [0,∞]. The intuition is that the more D_1 and D_2 overlap, the bigger $Eq(N_1, N_2)$; with $Eq(N_1, N_2)$ becoming infinite with $D_1 \equiv D_2$. $Eq(N_1, N_2)$ is normalized within [0,1]. The special case of $D_1 \equiv D_2$ is approximated to 1.

Let us first focus on the generalization relation. Given two nodes N_1 and N_2 and the related document sets D_1 and D_2, we introduce two additional sets: (i) the set of documents classified in the ancestor node of N_2, namely A_2, and (ii) the set of documents classified in the children nodes of N_1, namely C_1.

The generalization relationship holds when the first node has to be considered more general of the second node. Intuitively, this happens when the documents classified under the first node occur in the ancestor of the second node, or the documents classified under the second node occur in the subtree of the first node. Following this intuition we can formalize the generalization hypothesis as follows:

$$Mg(N_1, N_2) = \frac{|(A_2 \cap D_1) \cup (C_1 \cap D_2)|}{|D_1 \cup D_2|} \tag{35.2}$$

The specialization relationship hypothesis $Lg(N_1, N_2)$ can be easily formulated exploiting the symmetry of the problem.

The reference alignment for *TaxME* dataset is computed starting from Google, Yahoo! and Looksmart. The web directories hold many interesting properties: they are widely known, they cover overlapping topics, they are heterogeneous, they are large, and they address the same space of contents. All of this makes the working hypothesis of documents co-occurrence sustainable. The nodes are considered as categories denoted by lexical labels, the tree structures are considered as hierarchical relations, and the URLs classified under a given node are taken to denote documents. Table 35.1 summarizes the total amount of processed data.

Table 35.1: Number of nodes and documents processed in the *TaxME* construction process.

Web Directories	**Google**	**Looksmart**	**Yahoo!**
number of nodes	335.902	884.406	321.585
number of urls	2.425.215	8.498.157	872.410

Let us briefly summarize the five steps process used in the *TaxME* reference alignment construction.

Step 1 All three web directories are crawled, both the hierarchical structure and the web content;

Step 2 The URLs that do not exist in at least one web directory are discarded;

Step 3 The nodes with a number of URLs under a given threshold (10 in the experiment) are pruned;

Step 4 A manual selection is performed with the goal to restrict the assessment of the similarity metric to the subtrees concerning the same topic; 50 pairs of subtrees are selected;

Step 5 For each of the subtree pairs selected, an exhaustive assessment of correspondences holding between nodes is performed. This is done by exploiting the equivalence metric defined by Eq. 35.1 and the corresponding generalization (Eq. 35.2) and specialization metrics. The *TaxME* similarity metric is computed as the biggest out of the three metrics, namely as follows:

$$Sim_{TaxME} = max(Eq(N_1, N_2), Lg(N_1, N_2), Mg(N_1, N_2)) \tag{35.3}$$

The distribution of correspondences constructed using Sim_{TaxME} is depicted in Figure 35.2.

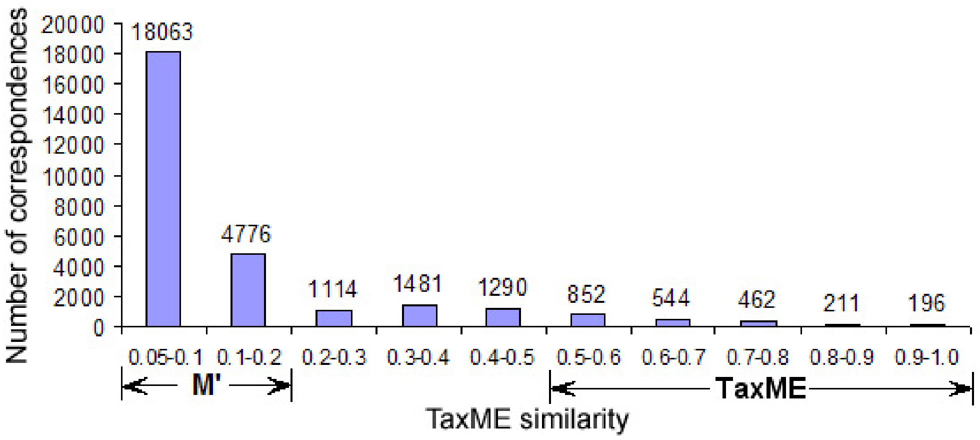


Figure 35.2: Distribution of correspondences according to the *TaxME* similarity metric.

Notice that Sim_{TaxME} is robust. The number of correspondences is in fact very stable and grows substantially, of two orders of magnitude, only with a value of the metric less than 0.1. As a pragmatic decision, the correspondences with Sim_{TaxME} above 0.5 are taken to constitute the reference alignment $TaxME$. As a result, $TaxME$ is composed from 2265 correspondences. Half of them are equivalence relationships and half are generalization relationships.

As depicted in Figure 35.3, $TaxME$ is an incomplete reference alignment since it contains only part of the correspondences in H. The key difference between Figure 35.3 and Figure 23.2 is the fact that a complete reference alignment (the area inside the dotted circle in Figure 35.3) is simulated by exploiting an incomplete one (the area inside the dashed circle in Figure 35.3).

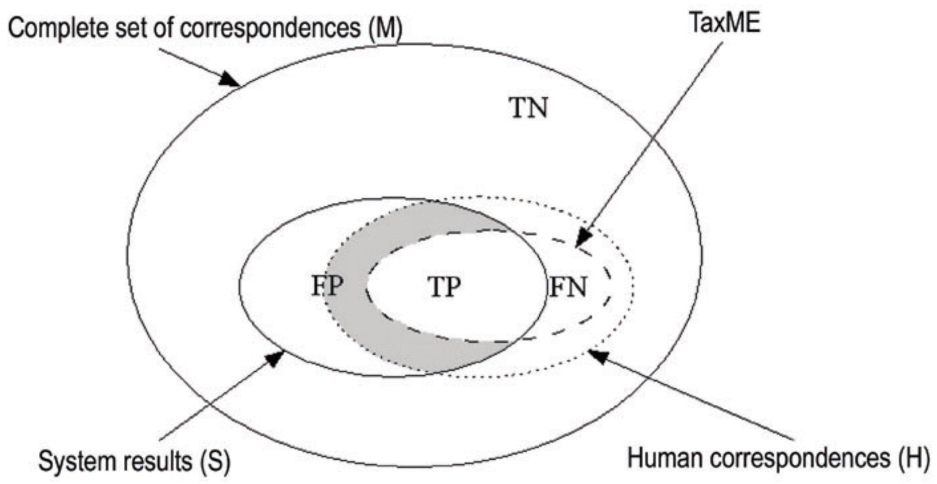

Complete set of correspondences (M)
TaxME
TN
FP
TP
FN
System results (S)
Human correspondences (H)

Figure 35.3: Alignment comparison using *TaxME*. TP, FN and FP stand for true positives, false negatives and false positives, respectively.

However, if we assume that *TaxME* is a good representative of H we can use definition of recall as defined in §23.1.3 (p.183) for its estimation. In order to ensure that this assumption holds a set of requirements have to be satisfied:

1. *Correctness*: $TaxME \subset H$ (modulo annotation errors).
2. *Complexity*: state of the art matching systems experience difficulties when run on *TaxME*.
3. *Discrimination capability*: different sets of correspondences taken from *TaxME* are hard for the different systems.
4. *Incrementality*: *TaxME* allows for the incremental discovery of the weaknesses of the tested systems[1].

As discussed in [Avesani *et al.*, 2005] *TaxME* satisfies these requirements. We have also evaluated the robustness of Sim_{TaxME}. We have randomly selected 100 correspondences in each of the intervals of Sim_{TaxME} values depicted in Figure 35.4 and manually evaluated their correctness. This resulted in a relatively small amount of manual work as we have analyzed around one thousand correspondences. The results are presented in Figure 35.4 and show that Sim_{TaxME} is robust, namely:

– It is very stable with a small percentage of incorrect correspondences for a very large range [0.3,1];

– The number of incorrect correspondences becomes substantial for very small values of Sim_{TaxME}, namely with threshold less than 0.1.

[1]We do not consider this property here as insignificant to our goals.

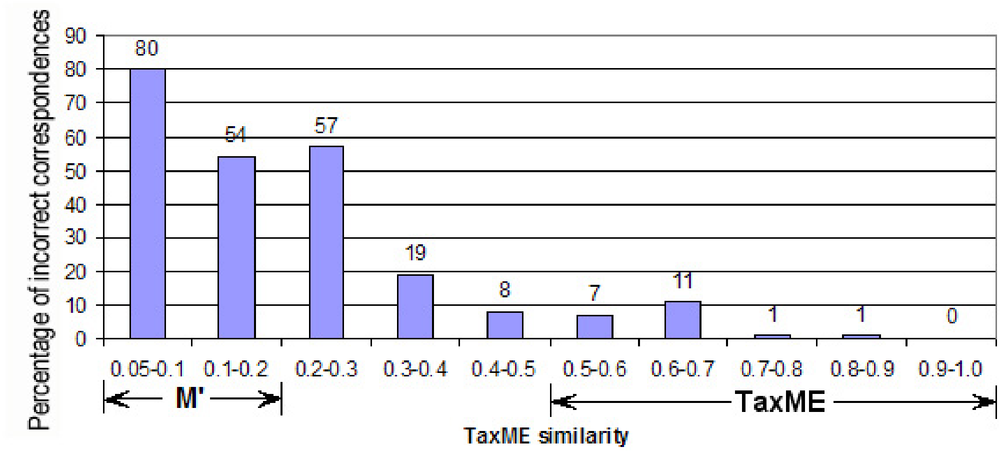


Figure 35.4: Distribution of incorrect correspondences. Each column is calculated evaluating 100 randomly selected correspondences.

35.1.2 A dataset for evaluating precision

As from §23.1.3 (p.183) in order to evaluate precision, we need to know FP, which in turn, as from Figure 23.2 (and Figure 35.3) requires to know H (reference alignment). However, computing H in the case of a large scale matching task requires an implausible human effort. We cannot either use an incomplete reference alignment composed from positive correspondences, i.e., $TaxME$. In this case, as shown in Figure 35.3, FP can not be computed. This is the case because $FP_{unknown} = S \cap (H - TaxME)$, marked as a gray area in Figure 35.3, is not known.

Our proposal here is to construct a reference alignment for the do2002a of both recall and precision, let us call it $TaxME_2$, defined as follows:

(35.4) $$TaxME_2 = TaxME \cup N_{T2}$$

N_{T2} is an incomplete reference alignment containing *only* negative correspondences (i.e., $N_{T2} \subset M - H$ in Figure 35.3). Of course $TaxME_2$ must be a good representative of M and therefore satisfy the requirements described in the previous section and satisfied by *TaxME*. Notice that the request of correctness significantly limits the size of N_{T2} since each correspondence has to be evaluated by a human annotator (i.e., $|N_{T2}| \ll |M - H|$). At the same time, N_{T2} must be big enough in order to be the source of meaningful results. Therefore, we require N_{T2} to be at least of the same size as $TaxME$, namely $|N_{T2}| \geq |TaxME|$.

N_{T2} is computed from the complete alignment set M in two macro steps. Let us first introduce them briefly and then discuss them in detail.

– *Step 1: Candidate correspondence selection.* The goal of this step is to select a set M' where $M' \subseteq M$ which contains a big number of "hard" negative correspondences.

– *Step 2: Negative correspondence selection.* The goal of this step is to filter all positive correspondences from M'. In order to achieve this goal M' is first pruned to the size that allows manual do2002a of the correspondences. Finally, the negative correspondences are manually selected from the remaining set of correspondences.

Candidate correspondence selection

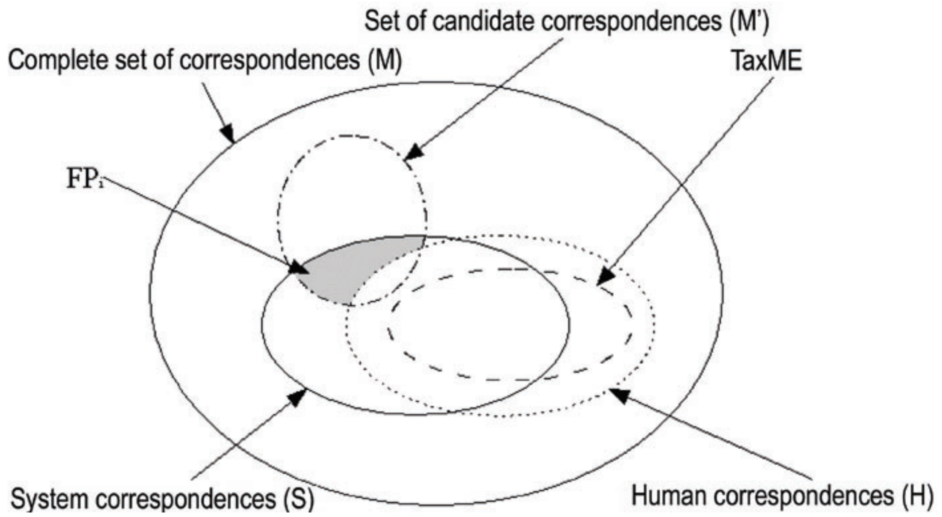

Set of candidate correspondences (M')
Complete set of correspondences (M)
TaxME
FPi
System correspondences (S)
Human correspondences (H)

Figure 35.5: Alignment sets in *TaxME 2*. Gray area stands for FP_i a set of FP produced by a matching system on M'.

The candidate set of correspondences M' is selected from M, as depicted in Figure 35.5. The goal of this step is to ensure that M' contains a big number of "hard" negative correspondences. Intuitively a "hard" negative correspondence is the correspondence with high value of similarity measure which is incorrect according to manual annotation. Taking into account the robustness of Sim_{TaxME} and also complexity and scalability requirements we have selected M' as the correspondences having Sim_{TaxME} values in the 0.05-0.2 range. As from Figure 35.2, this allowed us to obtain 18063+4776=22836 candidate correspondences.

Negative correspondence selection

The negative correspondence selection step is devoted to the computation of N_{T2}. The process is organized as follows:

– *Step 1: Matching system selection.* The goal of this step is to select a set of matching systems whose results are exploited for constructing N_{T2}. The set of the selected systems should be heterogeneous, i.e., the selected systems should make mistakes on different sets of correspondences. Thus, the selected systems have to be the representatives of the different classes of the existing matching techniques. This also prevents N_{T2} from being biased towards a particular class of matching solutions.

  As the result of this step, we have selected three different matching systems, namely COMA [Do and Rahm, 2002], Similarity Flooding (SF) [Melnik *et al.*, 2002] and S-Match (SM) [Giunchiglia *et al.*, 2004], see [Avesani *et al.*, 2005] for reasons of this choice.

– *Step 2: Computation of negative correspondences.* The goal of this step is to compute N_{T2} exploiting the results obtained by running the selected matching systems on M'. In particular, N_{T2} is computed from FP as $N_{T2} = \bigcup_i FP_i$, where FP_i stands for the FP produced by running the $i-th$ matching system on M'. The result of this exercise is depicted in Figure 35.5, where the gray area stands for FP_i. This construction schema ensures that N_{T2} will be hard for all existing systems and discriminative given that the set of matching systems evaluated on M' is representative and heterogeneous. An implicit constraint is that the number of FPs produced by each of the systems should be comparable. This prevents the existence of a bias towards a particular class of matching solutions. Notice that the computation of FP requires the human annotation of the systems results.

As the result of this step, we have executed COMA, SF and S-Match on M'. We also have manually evaluated the correspondences found by the systems and selected the FP from them. Notice that we have not distinguished among different semantic relations while evaluating the matching quality. Therefore, for example, the correspondence $A \sqsubseteq B$ produced by S-Match and $A_1 \equiv B_1$ produced by COMA have been considered as TP if $A \equiv B$ and $A_1 \sqsubseteq B_1$ are TP according to the human judgment. Finally, we have computed N_{T2} as the union of the FPs produced by the matching systems.

Table 35.2 provides a quantitative description of the content of N_{T2} and of the effort needed to build it. As from the first row of Table 35.2 the total number of annotated correspondences was 2553+2163+2151=6867. Notice that this is 6 orders of magnitude lower than the number of correspondences to be considered in the case of complete reference alignment. Notice also that the number of correspondences per system is very balanced, as required.

Table 35.2: Total number of correspondences and number of FP computed by COMA, SF and S-Match on M'.

	COMA	SF	SM
Found (S)	2553	2163	2151
Incorrect (FP)	870	776	781

Figure 35.6 shows how the FPs produced by the systems are partitioned. In particular, there are no FPs found by SM, COMA and SF, or even by SM and COMA together. There are the small intersections between the FPs produced by SM and by SF (0.1%) or by COMA and by SF (2.3 %). These results justify our assumption that all 3 systems belong to different classes.

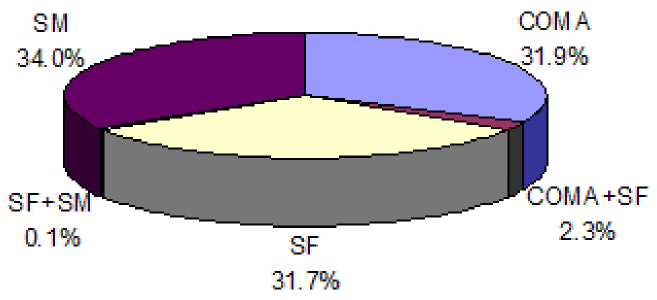


Figure 35.6: Partitioning of the FPs computed by COMA, SF and S-Match on M'.

The final result is that N_{T2} consists of 2374 correspondences. Notice that the size of N_{T2} is not equal to the sum of the FPs reported in the second row of Table 35.2 since, as from Figure

35.6, there is some intersection among these sets. The union of N_{T2} with $TaxME$ has allowed us to compute a reference alignment $TaxMe_2$, in turn, allowing for the do2002a of both recall and precision, of 2265+2374=4639 correspondences.

35.1.3 Test set summary

The data set exploited in the directory matching task was constructed from Google, Yahoo and Looksmart web directories following the methodology described in the previous subsections. The dataset is presented as taxonomies where the nodes of the web directories are modeled as classes and classification relation connecting the nodes is modeled as rdfs:subClassOf relation.

There were proposed three representations of this test case:

- one matching task between two taxonomies of 10^3 categories (full test set),
- one matching task between two taxonomies of 10^2 categories (10% test set), and
- 4639 matching tasks between two paths of around 10 categories (unit test sets).

The first data set incorporates the matching tasks involved in the unit tests which also correspond to the reference set. The second data set guarantees to contain 10% of these unit tests.

The reference alignment is composed of two parts:

- Representative subset of complete reference alignment ($P \subseteq R$). It contains the positive correspondences, i.e., the correspondences that hold for the matching task.
- Representative subset of negative correspondences ($N \subseteq \bar{R}$), i.e., the correspondences that do not hold for the matching task.

The reference alignment is composed of 2265 positive and 2374 negative correspondences. Therefore, the matching unit test set corresponds to 2265+2374=4639 tasks of finding a relation holding between paths in the web directories modeled as subclass hierarchies.

35.1.4 Results

Approximate precision, recall and F-measure of the systems on the directory dataset are presented in Figure 35.7, 35.8 and 35.9, respectively. Given an alignment A and the set P and N of positive and negative correspondences, approximate precision and recall are computed as follows:

$$AP(A, P, N) = \frac{|P \cap A|}{|P \cap A| + |N \cap A|} \qquad AR(A, P) = \frac{|P \cap A|}{|P|}$$

These formula, especially that of AP, generalize precision and recall by not taking the whole set of valid correspondences as reference alignment. They are called approximate precision and recall because when $P = R$ and $N = \bar{R}$ (R is the reference alignment), they correspond to precision and recall. How this is an accurate approximation of precision and recall heavily depends (as from the previous sections) on the choice of P and N.

Similarly to OAEI-2005, seven matching systems were evaluated on the dataset. However, only one of them (Falcon) participated in both do2002as. In OAEI-2006, the systems in general demonstrated better results than in OAEI-2005. The average approximate recall of the systems

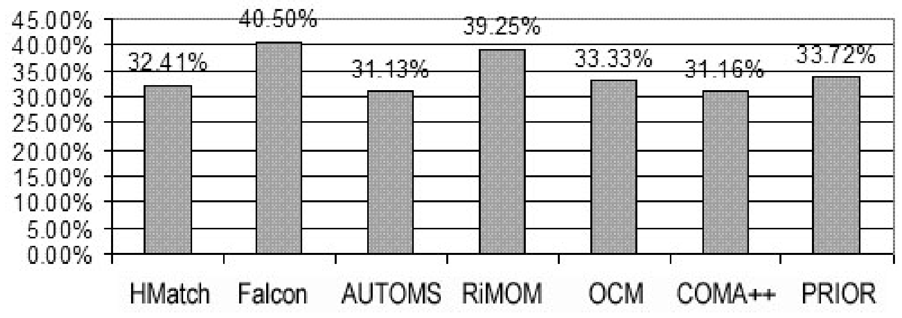


Figure 35.7: Approximate precision for web directories matching task.

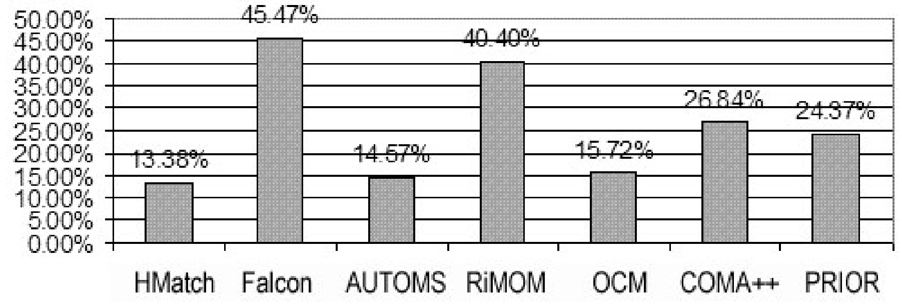


Figure 35.8: Approximate recall for web directories matching task.

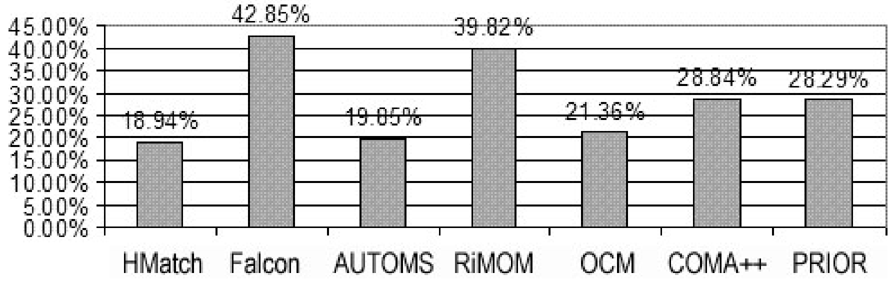


Figure 35.9: Approximate F-measure for web directories matching task.

increased from 22.23% to 25.82%. The highest approximate recall (45.47%) was demonstrated by the Falcon system what is almost a 50% increase with respect to its result of 2005 (31.17%).

Despite this progress the dataset remains difficult for the matching systems. The maximum and average values for approximate precision (40.5% and 34.5%), approximate recall (45.47% and 25.82%) and approximate F-measure (42.85% and 28,56%) are significantly lower than corresponding real values in benchmark tests for example.

Partition of positive and negative correspondences according to the systems results are presented in Figure 35.10 and Figure 35.11.

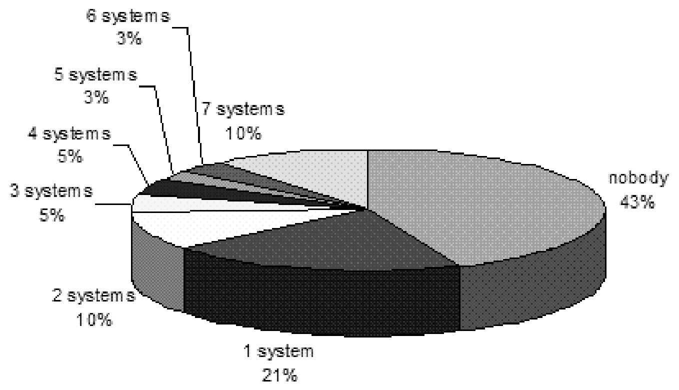


Figure 35.10: Partition of the systems results on positive correspondences.

Figure 35.10 and Figure 35.11 show that 43% of positive correspondences have not been found by any of the systems. At the same time 22% of negative correspondences were found by all the matching systems, i.e., all the matching systems mistakenly returned them as positive ones. Moreover, only 10% of positive correspondences were found by all the matching systems.

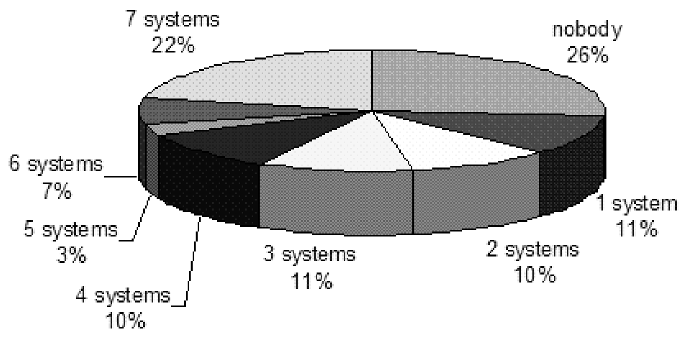


Figure 35.11: Partition of the systems results on negative correspondences.

35.1.5 Discussion

Six out of seven systems that participated in the do2002a presented their results only for one of the dataset representations, namely for the representation composed of 4639 node matching tasks. Only one system (HMatch) presented the results also for the other representations. Since, the other tasks were proposed in order to test the scalability of the approaches, this can be interpreted as a sign of poor scalability of the systems participating in the do2002a.

Blind do2002a offered for some of the systems a possibility to improve their final results after preliminary result disclosure. For example, the final results of the COMA++ and PRIOR matching systems were slightly lower than their preliminary results. The final F-measure of COMA++ dropped from 32.56% to 28.84% while F-measure of PRIOR dropped from 28.32% to 28.29%.

35.2 The food test case

The food test case aims at providing a realistic task for ontology matchers in the domain of thesauri. First, we introduce the test case (§35.2.1). Then, we outline the do2002a procedure (§35.2.2) and present test case results for the participated systems (§35.2.3). Finally, we discuss general issues that limit matching systems to produce better results on this test case (§35.2.4).

35.2.1 Test set

The thesauri used for this task are the Food and Agriculture Organization of the United Nations[2] (FAO) AGROVOC thesaurus[3] and the United States National Agricultural Library (NAL) Agricultural Thesaurus (NALT)[4]. Both thesauri were supplied unaltered to the participants in their native SKOS format[5]. An OWL-Lite translation made by Wei Hu was also supplied to the participants.

AGROVOC This thesaurus (version: May 2006) consists of 28.174 descriptor terms (i.e., preferred terms) and 10.028 non-descriptor terms (i.e., alternative terms). This version of AGROVOC is multilingual. It is available in ten languages, including English, French, Spanish, Arabic, Chinese, Portugese, Czech, Japanese, Thai, and Slovak. The AGROVOC thesaurus is used to index a multitude of data sources all over the world, one of which is the AGRIS/CARIS[6] literature reference database. A fragment of AGROVOC is shown in Figure 35.12 on the left side.

NALT This thesaurus (version: 2006) consists of 41.577 descriptor terms and 24.525 non-descriptor terms. It is monolingual and is available in English. The NALT thesaurus is used to index, amongst others, the AGRICOLA[7] literature reference database of the USDA[8], various data sources of the Agriculture Network Information Center[9] (AgNIC) and the Food Safety Re-

[2]`http://www.fao.org`
[3]`http://www.fao.org/agrovoc`
[4]`http://agclass.nal.usda.gov/agt/agt.shtml`
[5]`http://www.few.vu.nl/`$^{\text{sim}}$`wrvhage/oaei2006`
[6]`http://www.fao.org/agris`
[7]`http://agricola.nal.usda.gov`
[8]`http://www.usda.gov/wps/portal/usdahome`
[9]`http://www.agnic.org`

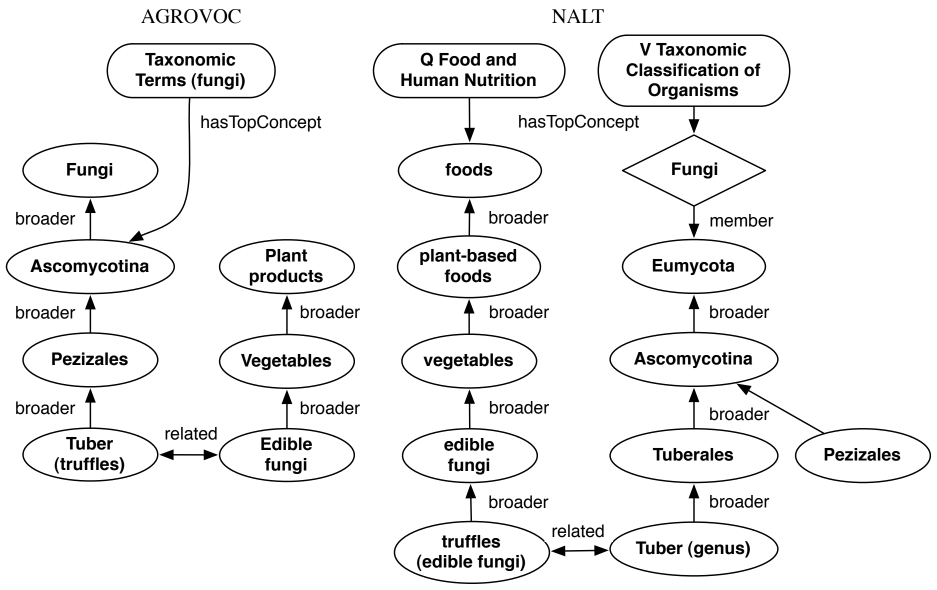


Figure 35.12: The concept of truffles in AGROVOC and NALT.

search Information Office[10] (FSRIO). A fragment of NALT is shown in Figure 35.12 on the right side.

SKOS We chose to use SKOS[11] because it closely follows "traditional" thesauri formats, such as the iso2709 format. Let us introduce some notation. We depict skos:Concepts as an oval filled with the skos:prefLabel text. In cases where we explicitly want to show skos:altLabel and skos:prefLabel we depict the skos:Concept as an oval filled with its URI, connected to boxes that represent its various labels. skos:Collections are depicted by diamond shapes and skos:ConceptShemes are shown as boxes with round sides.

SKOS mapping vocabulary For the correspondences we use the SKOS mapping vocabulary[12]. The participants were allowed to use the following relations: skosmap:narrowMatch, skosmap:exactMatch, and skosmap:broadMatch. The other relations and boolean combinators (skosmap:minorMatch, skosmap:majorMatch, skosmap:AND, skosmap:OR, skosmap:NOT) of the SKOS mapping vocabulary were not used in the do2002a.

35.2.2 Evaluation procedure

Precision In order to assess precision of the results within the time span of OAEI-2006 and given a limited number of assessors and man-hours we performed a sample do2002a. The samples were chosen to be representative of the type of topics covered by the thesauri and to be impartial to each participant and impartial to how much consensus amongst the participants there was

[10]`http://fsrio.nal.usda.gov`
[11]`http://www.w3.org/TR/swbp-skos-core-spec`
[12]`http://www.w3.org/2004/02/skos/mapping/spec`

about each correspondence (i.e., the "hardness" or "complexity" of the correspondence). This was accomplished with stratification.

Stratification of the matching results. We distinguished three categories of topics in the thesauri. Each of these required a different level of domain knowledge of the assessors: *taxonomical* concepts (plants, animals, bacteria, etc.), *biological and chemical* terms (structure formulas, terms from generics, etc.), and *miscellaneous*, the remaining concepts (geography, agricultural processes, etc.). The correspondences were partitioned into strata corresponding to these topics. Under the authority of taxonomists at the USDA the taxonomical stratum was assessed completely using the strict rules that apply to the naming scheme of a taxonomy. These rules state that two taxonomical concepts are considered exact matches if and only if the skos:prefLabel of one concept is literally the same as either the skos:prefLabel or the skos:altLabel of the other concept. From the latter two strata we took samples that were assessed by two groups: (i) a group of domain experts from the USDA and the FAO, and (ii) a group of computer scientists at the EKAW conference[13]. The size of the strata and the size of the assessed samples are shown in Table 35.3. The samples were selected in such a way that each participant has an equal share in the sample. Also, there is an equal number of correspondences in the sample for each "agreement level", where "agreement level" is defined as the set of correspondences returned by n systems. This means that from each of the columns in Table 35.4 (p.273) marked 1–5, and from each row that represents a system, we took a sample of equal size[14].

Table 35.3: Sizes of the strata and of the samples from those strata that were assessed to evaluate precision.

Stratum topic	Stratum size (N_h)	Sample size (n_h)
Taxonomical	18.399	18.399
Bio/chem	2.403	250
Miscellaneous	10.310	650
All topics	31.112	

Assessment tool for precision. For the assessment of precision we used a tool developed by TNO[15], see Figure 35.13 for a screen shot. This tool reads a set of correspondences in the common format for alignments and outputs a web form that is used by judges to assess the correspondences. The results of the form are submitted to the organizer of the food task. The assessment process of a correspondence follows three steps (see also Figure 35.13).

1. The judge decides if the relation specified above the arrow between the two (green) boxes holds between the two concepts displayed in bold face. If the relation holds (s)he skips step 2 and proceeds to step 3 directly, otherwise step 2 is performed.

2. The judge tries to specify an alternative relation, either by changing the relation type, or the concepts. If possible (s)he selects "exactMatch" and specifies the proper concepts be-

[13]`http://ekaw.vse.cz/`

[14]The samples can be downloaded from `http://www.few.vu.nl/`$^{\text{sim}}$`wrvhage/oaei2006/gold_standard`.

[15]`http://www.tno.nl/index.cfm?Taal=2`

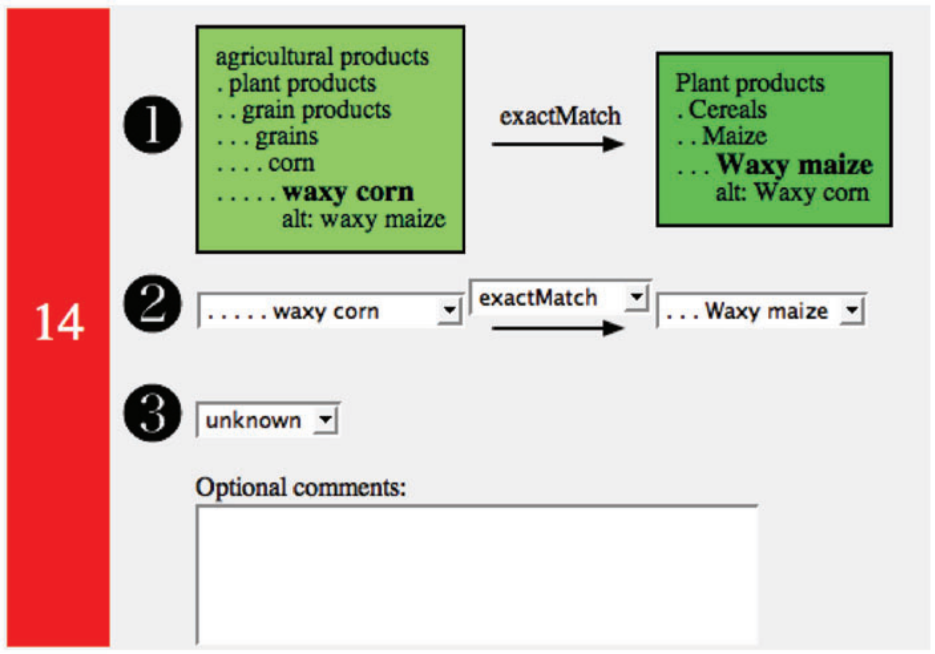


Figure 35.13: Screen shot of the assessment tool used to evaluate precision. It shows the 14th correspondence relation from a sample set of correspondences: nalt:'waxy corn' skosmap:exactMatch agrovoc:'Waxy maize'.

tween which the "exactMatch" relation holds. Otherwise, (s)he selects "broadMatch" or "narrowMatch" and specifies the proper concepts between which that relation holds.

3. The judge changes the default value ("unknown") of the assessment into either "true" or "false". If the relation holds and step 2 was skipped, then (s)he selects "true". If the relation does not hold, but if (s)he successfully selected an alternative relation (at step 2) that does hold, (s)he also selects "true". If the relation does not hold and no correct alternative could be found at step 2, (s)he selects "false".

Finally, if the judge wishes to document the decision taken (s)he may fill in the box called "Optional comments" at the bottom of the assessment form.

Inter-judge agreement. The agreement between the group of domain experts and the group of computer scientists was 72%. The computer scientists were less likely to judge a correspondence to be correct than the domain experts. They judged 78% of the sample correspondences to be "true", while the domain experts judged 85% to be "true". The effect of this is that the estimated precision based on the computer scientists' work, as compared to that based on the domain experts' work, will be relatively low. We have no data on the inter-judge agreement within these groups.

Significance testing. As a significance test on precision scores of the systems we used the Bernoulli distribution. Precision of system A (denoted by P_A) can be considered to be significantly greater than precision of system B (denoted by P_B), if their estimated values, $\hat{P}_A$ and $\hat{P}_B$

are far enough apart. In general, based on the Bernoulli distribution, this is the case when the following formula holds:

$$|\hat{P}_A - \hat{P}_B| > \frac{2}{n}\sqrt{\big(\hat{P}_A(1-\hat{P}_A)\big)^2 + \big(\hat{P}_B(1-\hat{P}_B)\big)^2}$$

This significance test was used to determine which of the systems performs best for each of the topic strata.

We calculated these estimations based on three strata. This allows us to distinguish smaller differences in the results than by simple random sampling by combining the results of the strata. We denote the estimated precision of system A on stratum h as $\hat{P}_{A,h}$, the size of stratum h as N_h, and the size of the sample from stratum h as n_h (see also Table 35.3). We can conclude that system A performs significantly better than system B when the following formula holds:

$$|\hat{P}_A - \hat{P}_B| > \frac{2}{N}\sqrt{\Big(\sum_{h=1}^{L}\hat{P}_{A,h}(1-\hat{P}_{A,h})(\frac{N_h}{n_h}-1)\Big)^2 + \Big(\sum_{h=1}^{L}\hat{P}_{B,h}(1-\hat{P}_{B,h})(\frac{N_h}{n_h}-1)\Big)^2}$$

This significance test was used to determine which of the systems performs best for all the strata.

Recall Assessing the recall within the time span of OAEI-2006 was not feasible, so we estimated it based on four sample sub-hierarchies of the thesauri: (i) all oak trees (everything under the concept representing the Quercus genus), (ii) all rodents (everything under Rodentia), (iii) Geographical concepts of Europe, (iv) everything under the NALT concept animal health and all AGROVOC concepts that have correspondences to these concepts and their sub-concepts. These four samples respectively have sizes 41, 42, 74, and 34. Around 30% of the correspondences were broadMatch and narrowMatch, the rest was exactMatch. Currently, the sample for recall is extended for OAEI-2007[16].

Assessment tool for recall. To create the samples we used the AIDA Thesaurus Browser. That is a SKOS browser that supports parallel browsing of two thesauri, concept search, correspondence traversal, and the addition, change and removal of correspondences of the SKOS mapping vocabulary, see Figure 35.14 for a screen shot. This tool was developed by TNO[17] in the context of the VL-e project[18].

A preliminary version of the recall samples were made at the Vrije Universiteit Amsterdam and was verified and extended by domain experts at the the FAO and USDA to produce the final recall samples[19]. The guidelines used to make the correspondence were the following:

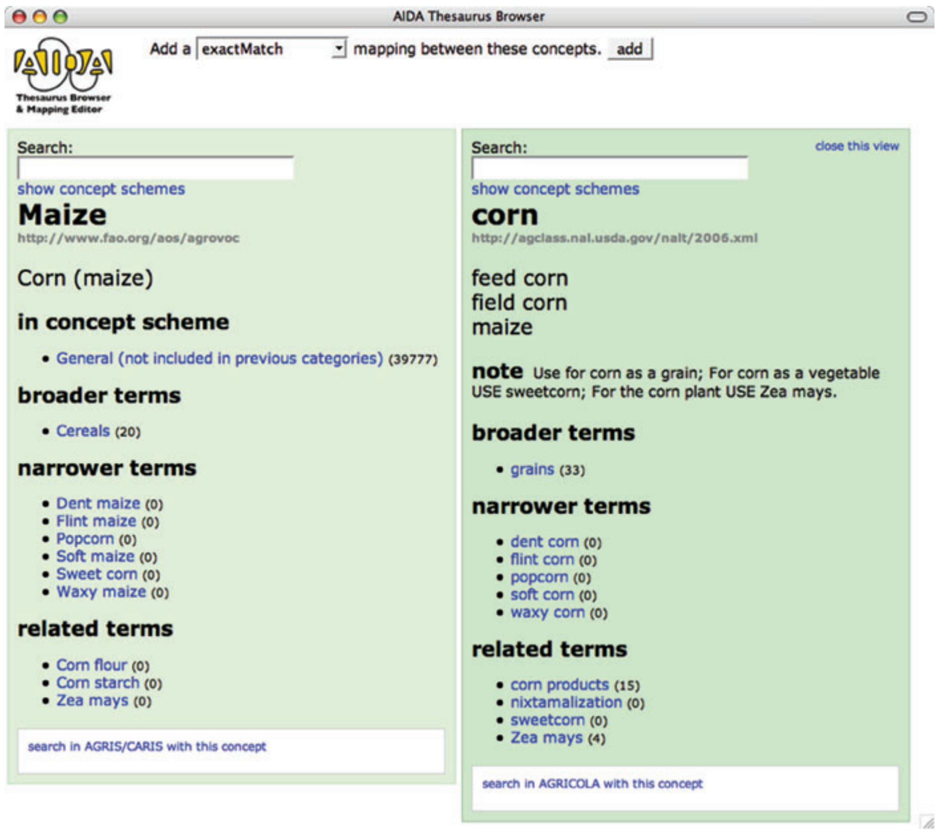
1. Starting from AGROVOC, identify a skosmap:exactMatch for every concept in the sample. If this is impossible, try to find a skosmap:narrowMatch or skosmap:broadMatch. Always choose the broader concept of these correspondences as narrow as possible and the narrower concept as broad as possible.

[16] http://oaei.ontologymatching.org/2007

[17] http://www.tno.nl/index.cfm?Taal=2

[18] http://www.vl-e.nl/frame_home.htm

[19] The samples can be downloaded from http://www.few.vu.nl/simwrvhage/oaei2006/gold_standard.



Add a exactMatch mapping between these concepts.

Search:
show concept schemes

Maize
http://www.fao.org/aos/agrovoc

Corn (maize)

in concept scheme
- General (not included in previous categories) (39777)

broader terms
- Cereals (20)

narrower terms
- Dent maize (0)
- Flint maize (0)
- Popcorn (0)
- Soft maize (0)
- Sweet corn (0)
- Waxy maize (0)

related terms
- Corn flour (0)
- Corn starch (0)
- Zea mays (0)

search in AGRIS/CARIS with this concept

Search:
show concept schemes

corn
http://agclass.nal.usda.gov/nalt/2006.xml

feed corn
field corn
maize

note Use for corn as a grain; For corn as a vegetable USE sweetcorn; For the corn plant USE Zea mays.

broader terms
- grains (33)

narrower terms
- dent corn (0)
- flint corn (0)
- popcorn (0)
- soft corn (0)
- waxy corn (0)

related terms
- corn products (15)
- nixtamalization (0)
- sweetcorn (0)
- Zea mays (4)

search in AGRICOLA with this concept

Figure 35.14: Screen shot of the AIDA Thesaurus Browser, which was used to create correspondence samples for the do2002a of recall.

2. Investigate the surrounding concepts of the target concept in NALT. If the surrounding concepts are still on the topic for the sample, try to match this concept “back” to AGROVOC using skosmap:exactMatch. If this is impossible, try to find a skosmap:narrowMatch or skosmap:broadMatch.

Significance tests. We exploited the same significance tests as used for precision.

35.2.3 Results

Five participants took part in the OAEI-2006 food test case: South East University (Falcon) [Jian *et al.*, 2005], University of Pittsburgh (PRIOR) [Mao and Peng, 2006], Tsinghua University (RiMOM) [Li *et al.*, 2006], University of Leipzig (COMA++) [Maßmann *et al.*, 2006], and Universitá degli Studi di Milano (HMatch) [Castano *et al.*, 2006a]. Each team provided between 10.000 and 20.000 correspondences. This amounted to 31.112 unique correspondences in total. All of these correspondences were of type skosmap:exactMatch. None of the systems was able to detect skosmap:broadMatch or skosmap:narrowMatch correspondences. There was a “high agreement” between the best three systems, RiMOM, Falcon, and HMatch, see Table 35.4. This table also indicates that there is a relatively large set of “easy” correspondences that are recognized by all systems.

Table 35.4: Distribution of the systems' results. It shows the number of correspondences returned by each system and how many correspondences are returned by n out of 5 systems.

	# Correspondences	# Correspondences shared amongst n systems				
System	returned	1	2	3	4	5
RiMOM	13,975	868	1,042	2,121	4,389	5,555
Falcon	13,009	642	419	1,939	4,400	5,555
PRIOR	11,511	1,543	1,106	676	2,631	5,555
COMA++	15,496	11,610	1,636	629	2,028	5,555
HMatch	20,001	7,000	981	2,045	4,420	5,555
all systems	31,112	21,663	2,592	2,470	4,467	5,555

The taxonomical parts of the thesauri accounted for the largest part of the correspondences, 59% of all submitted correspondences. The more difficult correspondences that required lexical normalization, such as structure formulas, and relations that required background knowledge, such as many of the relations in the miscellaneous domain, accounted for a smaller part of the correspondences. This caused systems that did well at the taxonomical correspondences to have a great advantage over the other systems. The Falcon system performed consistently best at the largest two strata, taxonomical and miscellaneous, and thus achieved high precision, see Table 35.5.

Table 35.5: Precision results based on sample do2002a. "⋆" indicates the best results.

Precision for	RiMOM	Falcon	PRIOR	COMA++	HMatch
taxonomical	82%	83%⋆	68%	43%	48%
bio/chem	85%⋆	80%	81%	76%	83%
miscellaneous	78%	83%⋆	74%	70%	80%
all topics	**81%**	**83%**⋆	**71%**	**54%**	**61%**

All systems only returned skosmap:exactMatch correspondences. This means that the recall of all systems was limited to 71%. For example, RiMOM achieved 50%, while it could achieve 71%, see Table 35.6. The RiMOM system managed to discover more good results than the Falcon system on the four small sample recall bases (at the expense of precision). Notice that the recall was assessed on a small set of examples, therefore we can only draw certain conclusions based on the precision results. Table 35.7 summarizes tentative F-measure results.

Table 35.6: Tentative estimation of recall based on sample do2002a.

Recall for	RiMOM	Falcon	PRIOR	COMA++	HMatch
all relations	**50%**	**46%**	**45%**	**23%**	**46%**
only exactMatch	71%	65%	64%	33%	65%

Table 35.7: Tentative estimation of F-measure based on sample do2002a.

F-measure for	RiMOM	Falcon	PRIOR	COMA++	HMatch
all rel. & top.	**62%**	**59%**	**55%**	**33%**	**53%**

A potential user of ontology matching systems does not necessarily have to limit himself/herself to only one matching system. Simple ensemble methods such as majority voting can improve precision. To give an impression of this we list the average precision of the different "agreement levels" in Table 35.8. For agreement level 4 and 5 (i.e., the correspondences that were returned by 4 out of 5 systems or all of the systems) precision is significantly higher than for the best system by itself, Falcon in this case. Nearly all of the 5,555 correspondences found in this way are correct. Obviously, these are the "easy" correspondences. Whether they are useful or not useful depends on the application of the correspondences and remains a topic for future research.

Table 35.8: Consensus: average precision of the correspondences returned by a number of systems.

correspondence found by # systems	1	2	3	4	5
average precision	6%	35%	67%	86%	99%
# correspondences	21.663	2.592	2.470	4.467	5.555

35.2.4 Discussion

Let us discuss a number of issues that limit the performance of matching systems. Some of these issues are technical and easy to solve, while others are more fundamental and represent challenges to be tackled in a longer term.

Inappropriate "spelling correction". Incorrect matches such as nalt:patients skosmap:exactMatch agrovoc:Patents and nalt:aesthetics skosmap:exactMatch agrovoc:anaesthetics are caused by inappropriate spelling correction. In general, spelling tolerance in thesauri is not very effective, but if it is applied nonetheless it should only be applied when there is no exact literal match. For example, there is a concept representing "patients" in both thesauri. Recognizing this should trigger a matching system to refrain from suggesting a correspondence to "patents".

Labels following naming schemes. Labels often follow naming schemes. Real-life ontologies often use more than one naming scheme. Both AGROVOC and NALT have a large section on biology. The labels of these concepts follow the Linnaeic system of species names. Concepts in other sections of the thesauri (e.g., the sections on geography) follow different schemes. It is vital that lexical matchers recognize that different naming schemes require different matching rules. One of the common matching rules is "suffix" matching. For instance, "lime stone" and "sand stone" can be found similar using the suffix matcher. In fact, they are both kinds of "stone". However, two terms from the Linnaeic system that end in the same word, such as "Quercus pubescens" (a tree) and "Ibacus pubescens" (a crustacean) are completely dissimilar. Failing to recognize that the Linnaeic system needs also prefix matching and not only suffix matching can lead to many wrong correspondences. The bold arrow in Figure 35.15 indicates this wrong correspondence.

"Use" and "use for" modeled with skos:altLabel. When use is modeled using skos:altLabel the difference between synonyms, obsolete terms, and acknowledgment of lack of detail disappears. For example, in Figure 35.16 AGROVOC does not include detailed descriptors for the concept nalt:Sigmodon. In fact, a few levels of taxonomical distinctions are left out. The skos:altLabel "Sigmodon" is added to indicate this omission. It indicates that users that desire to refer to sigmodons should use the agrovoc:c_6633 concept, that symbolizes all rodents. A matcher without

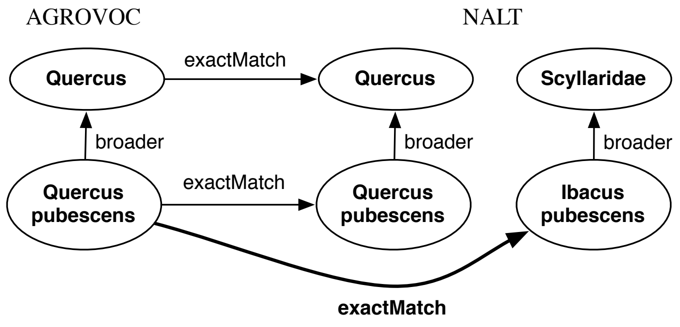


Figure 35.15: Failing to recognize the naming scheme can lead to wrong correspondences.

prior knowledge about this modeling decision cannot distinguish this from synonymy represented with skos:altLabel. This will cause most systems to conclude that there is a skosmap:exactMatch between agrovoc:c_6633 and nalt:Sigmodon, while the proper relation between these concepts is a skosmap:narrowMatch.

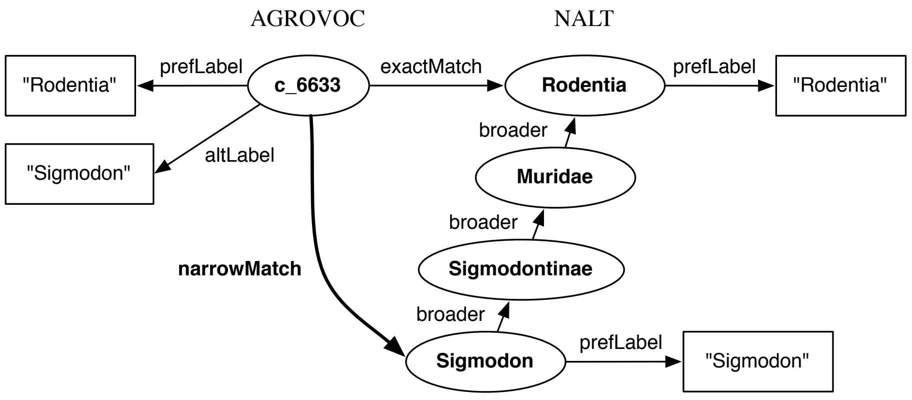


Figure 35.16: use modeled with skos:altLabel in AGROVOC.

Colloquial names and scientific names. A delicate problem is that of colloquial versus scientific names for the same species. Take the example illustrated in Figure 35.17 of gerbils with the scientific name “Gerbilinae”. In NALT, the two types of names each have their own hierarchy. In AGROVOC these two are combined, because they both refer to the same actual species. This leads us to believe that in this case there should be skosmap:exactMatch correspondences to both hierarchies in NALT. We created the do2002a samples for recall based on this assumption. Whether it is the proper treatment depends on the application of the correspondences. The different names can symbolize different views of the concepts or refer to the same extension. A problem similar to this example occurs with colloquial names, scientific names, and structure formula’s of chemicals.

Clashing senses. Let us consider “Ireland” and the “British Isles”. The British Isles can be partitioned in two ways: the Irish Republic and the United Kingdom; or Ireland and the other islands of the British Isles, which all belong to the United Kingdom. If the distinction is made between Ireland and the United Kingdom, the most obvious interpretation is the former partition.

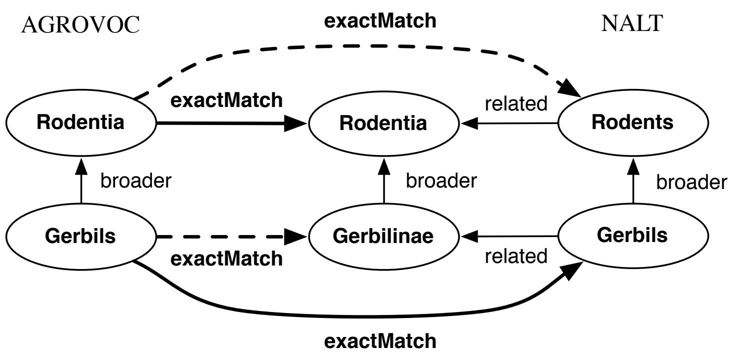


Figure 35.17: Separate hierarchies for colloquial names and scientific names.

This is due to the fact that most people assume sibling concepts to be disjoint. The lack of a broader relation between agrovoc:Northern Ireland and agrovoc:Ireland supports this. Another common assumption is that narrower concepts are strictly narrower than (i.e., not equivalent to) their parents. This means that the existence of the concept nalt:Irish Republic makes people assume that nalt:Ireland refers to the entire island. The narrower concept Northern Ireland confirms this. This means that agrovoc:Ireland should be equivalent to nalt:Irish Republic. In this case, this problem could be solved by adding OWL statements that proclaim siblings to be disjoint and broader concept to be not equivalent to narrower concepts. This kind of approach, however, is likely to cause more harm than good in the entire thesaurus. Thesaurus concepts are inherently vague and such a strict interpretation often causes unintentional inconsistencies. A technique that uses the added axioms as heuristics might be more suitable.

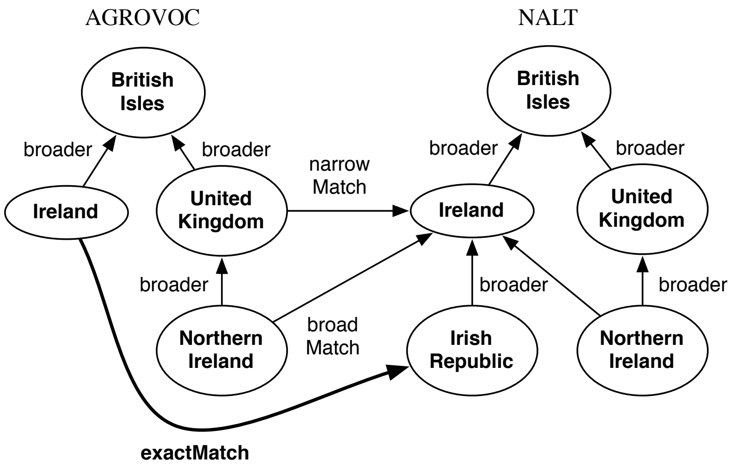


Figure 35.18: Concepts representing different senses of a term.

No direct evidence in the thesauri for a correct correspondence. In many cases it is simply impossible to find certain correspondences without resorting to external knowledge sources, such as a third ontology, concrete domain reasoning, text mining, or traditional knowledge acquisition. For example, in Figure 35.19 Western Europe is a named geographical region, but the skos:broader

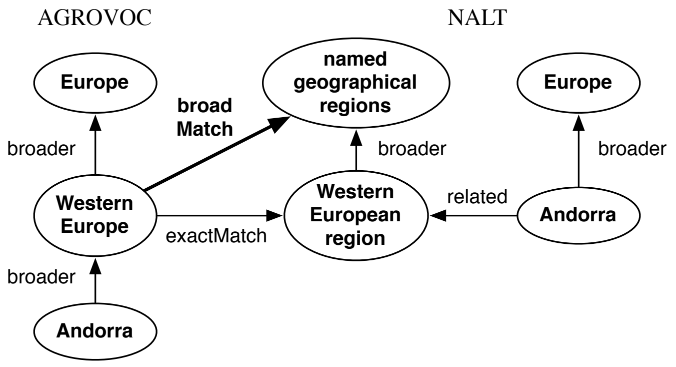


Figure 35.19: The is no evidence in the thesauri for this skosmap:broadMatch.

relation between nalt:Western European region and nalt:named geographical regions alone is not enough evidence to suggest this. For example, AGROVOC contains no concepts that are lexically similar to the latter NALT concept.

Chapter 36

The conference track and consensus workshop

The conference test set introduces matching several ontologies together as well as a consensus workshop aiming at studying the elaboration of consensus when establishing reference alignments. First we present the underlying test set (§36.1). Then, we provide analysis of the do2002a results via manual labeling (§36.2), logical reasoning (§36.3), the consensus building workshop (§36.4), and, ultimately, by means of pattern-aware data mining (§36.5). Finally, we outline some related works and summarize the major findings of this chapter (§36.6).

The material of this chapter has been published in [Šváb *et al.*, 2007].

36.1 The *OntoFarm* collection

The motivation for initiating the creation of the *OntoFarm*[1] collection (in Spring 2005) was the lack of 'manageable' material for testing ontology engineering (especially, matching) techniques. As underlying domain, we chose that of *conference organization*—among other, for the following reasons:

– Most ontology engineers are academics who themselves submit and review papers and organize conferences: there is zero overhead of acquiring the domain expertise.

– Organization of a conference shares some aspects with (heavier-weighted) business activities: access restrictions, hard vs. soft constraints, temporal dependencies among events, evolution of the meaning of concepts in time, etc. There is also a wide range of supporting software tools covering various aspects of conference organization. Their domain assumptions can also be captured using ontologies (specific for each system). The process of matching the requirements of conference organizers with the capacities of such tools is analogous with that of matching the requirements of a business with the capacities of an off-the-shelf enterprise information system.

– In many cases, even the underlying instance data could be obtained, since legal restrictions are typically not as strong as, e.g., in business or medicine.

[1]`http://nb.vse.cz/`$^{\rm sim}$`svabo/oaei2006`

In OAEI-2006 the conference test case consisted of ten OWL-DL ontologies, typically of the size of 30–80 concepts and 30–60 properties, see Table 36.1. Most ontologies were equipped with DL axioms of various kinds. Six among the ontologies were derived from different conference organization support tools (for the review process, registration, etc.), using their documentation and experiments with installed tools ('tool' ontologies); two of them are based on the experience of people with personal participation in conference organization ('insider' ontologies); finally, two of them are merely based on the content of web pages of concrete conferences ('web' ontologies). The ontology designers (partly students of a course on Knowledge Modelling and partly experienced knowledge engineers) did not interact among themselves. This should guarantee that, although the ontologies themselves are to some degree artificial (their development not being driven by an application need), their heterogeneity was introduced in a natural way, that possibly simulating the heterogeneity of ontologies developed by different communities in the real world.

Table 36.1: Characteristics of the *OntoFarm* ontologies.

Name	Type	Number of Classes	Number of Properties	DL expressivity
EKAW	Insider	77	33	$\mathcal{SHIN}(\mathcal{D})$
SOFSEM	Insider	60	64	$\mathcal{ALCHIF}(\mathcal{D})$
SIGKDD	Web	49	28	$\mathcal{ELI}(\mathcal{D})$
IASTED	Web	140	41	$\mathcal{ALCIF}(\mathcal{D})$
Confious	Tool	57	57	$\mathcal{SHIN}(\mathcal{D})$
PCS	Tool	23	38	$\mathcal{ELUIF}(\mathcal{D})$
OpenConf	Tool	62	45	$\mathcal{ALCIO}(\mathcal{D})$
ConfTool	Tool	38	36	$\mathcal{SIF}(\mathcal{D})$
CRS	Tool	14	17	$\mathcal{ALCIF}(\mathcal{D})$
CMT	Tool	36	59	$\mathcal{ALCIF}(\mathcal{D})$

36.2 Initial manual empirical do2002a

There were six participant groups to the OAEI-2006 conference track, with the following matching systems: AUTOMS, COMA++, OWL-CtxMatch, Falcon, HMatch and RiMOM. The alignments obtained were examined by the organizers, and each individual correspondence was assigned a label. Results from the initial do2002a phase consist in global statistics about the participants results, which more-or-less reflect their quality. Additional, finer-grained results were obtained at the consensus building workshop (§36.4).

The global statistics for each system amount to (among other):

- The distinction whether the correspondence is true/false or ranges between 0 and 1;
- Number of alignments;
- Number of individual correspondences labeled as correct vs. incorrect;
- Number of interesting correct correspondences, namely, those that were subjectively not so easy to identify at first sight (e.g., due to lack of string similarity);

– Number of correspondences that seemed to exhibit an interesting type of error (or problematic feature), specifically for: subsumption mistaken for equivalence, sibling concepts mistaken for equivalent ones, mutually inverse properties matched on each other, relation matched onto class;
– Precision and recall measures.

Additionally, some of the correspondences that were retained as worth discussing by both independent evaluators were then submitted to the consensus building workshop.

36.3 Empirical do2002a via logical reasoning

In addition to manual do2002a, we conducted an automatic analysis on a subset of the correspondences. Correspondences between class names in different ontologies were formalized in C-OWL [Bouquet *et al.*, 2003a] and the DRAGO system [Serafini and Tamilin, 2005] was used to determine whether the correspondences created by a particular system cause logical inconsistencies in one of the matched ontologies. C-OWL was chosen as basis for the do2002a, as its semantics is tuned towards describing correspondences between ontologies of the same domain; it solves some problems that occur when standard OWL is used for this purpose. A more detailed description of the approach can be found in [Meilicke *et al.*, 2006]. The analysis was performed on six ontologies with four matching systems, namely Falcon, OWL-CTXmatch, COMA++ and HMatch.

Table 36.2 shows the results of the reasoning-based analysis. The first column lists the systems under consideration. The second column lists the number of correspondences produced by the given system that make the target ontology inconsistent, the third column lists the average number of inconsistent concepts per correspondence, and the fourth column shows the overall precision of the alignment. Note that the precision only refers to correspondences between class names and therefore naturally differs from the numbers at the result report page. The precision has been determined by a manual investigation of the correspondences by three independent people (different from those doing almost the same task for the sake of the result report page). In cases of a disagreement the correctness of a correspondence was decided by a majority vote. It however turned out that there was little disagreement with respect to the correctness of correspondence. For only about 3% of the correspondences the result had to be determined by vote.

The results of this do2002a are useful in two ways. First of all, we can see from the numbers that a low number of inconsistent alignments is an indicator for the quality of correspondences (we also see that the actual number of concepts that become unsatisfiable is less relevant). The second benefit of this do2002a is the fact that the information about inconsistent concepts and

Table 36.2: Results of reasoning-based do2002a.

System	Inconsistent correspondences	Avg. number of inconsistent concepts	Overall precision
Falcon	4	1,5	89,7 %
OWL-CTXmatch	6	9,6	85,67 %
COMA++	12	2,2	67,7 %
HMatch	9	5,5	63,7 %

correspondences that caused these inconsistencies reveal obvious and also non-obvious errors in correspondences. Some examples of obviously incorrect correspondences produced by matching systems in the experiments are the following:

$$\begin{aligned} Document &= Topic \\ Decision &= Location \\ Reception &= Rejection \end{aligned}$$

The real benefit of this do2002a is its ability to find non-obvious errors in correspondences that can only be detected taking the position of the matched concepts in the concept hierarchy into account. In our experiments, we found a number of such errors. Examples include the following correspondences:

$$\begin{aligned} Regular\_Paper &= Regular \\ Reviewing\_event &= review \\ Main\_office &= Location \end{aligned}$$

In the case of the first correspondence, *Regular* actually denotes the regular participation fee as opposed to the early registration. The error in the second correspondence is caused by the fact that *Reviewing_event* represents the process of reviewing whereas review denotes the review document as such. The last correspondence is not correct, because the concept *Main_office* actually represents the main office as an organizational unit rather than a location. Such correspondences are candidates for a closer inspection in terms of a committee of experts that analyze the reason for the inconsistency and decide whether the problem is in the correspondence or in the ontologies.

36.4 Consensus building workshop

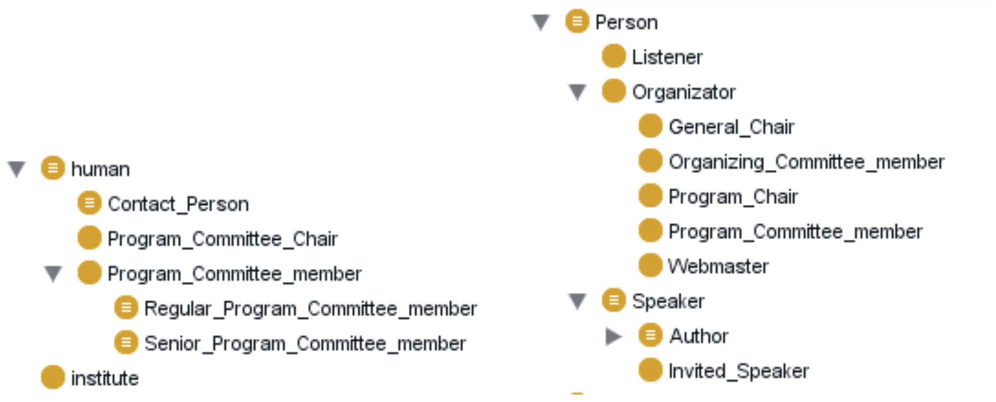
The idea of consensus building workshop was to discuss some interesting correspondences in detail. Such interesting correspondences are determined as a result of the manual and the automatic do2002a of the matching results, as shown above. In the case of the manual do2002a correspondences where the evaluators where in doubt or where they disagreed on the correctness of a correspondence are candidates for a consensus workshop. In the automatic do2002a, correspondences that have been shown to cause concepts in the matched ontologies to become inconsistent are such candidates, especially if the correspondences have been annotated as being correct in the manual do2002a. Often, a decision whether a correspondence is correct or not can be made quite easily in a committee of experts. In some cases, however, it turns out that deciding whether a correspondence is correct or not is far from being trivial. In particular, it turns out that sometimes a detailed analysis of the matched ontologies is necessary to come to a decision.

As far as arguments against and for individual correspondences are concerned, we experienced that *lexical* reasons of correspondence were first considered by the workshop participants. Then followed arguments with regard to the *context* of elements in question. This means consideration of certain neighborhood, subclasses and superclasses (in the case of properties, we can consider subproperties and superproperties). This can disclose different extensions of classes (especially through their subclasses). Also, properties related to classes were considered. As a last resort, *axioms* (more complex restrictions) were taken into account if they were present.

36.4.1 Examples of correspondences discussed

In the following, we focus on examples that illustrate the kinds of arguments used in the discussion and the insights gained.

Person vs. human: At first sight the equivalence between the concepts person and human looks rather intuitive, it is however not obvious that the two concepts have the same intended meaning in different ontologies. First of all, the concept person can be interpreted in a legal context in which it also refers to organizations. Further, when we look at the hierarchies of the different ontologies, we see that the concepts have completely different sets of subconcepts depending on the scope of the ontology, see Figure 36.1.

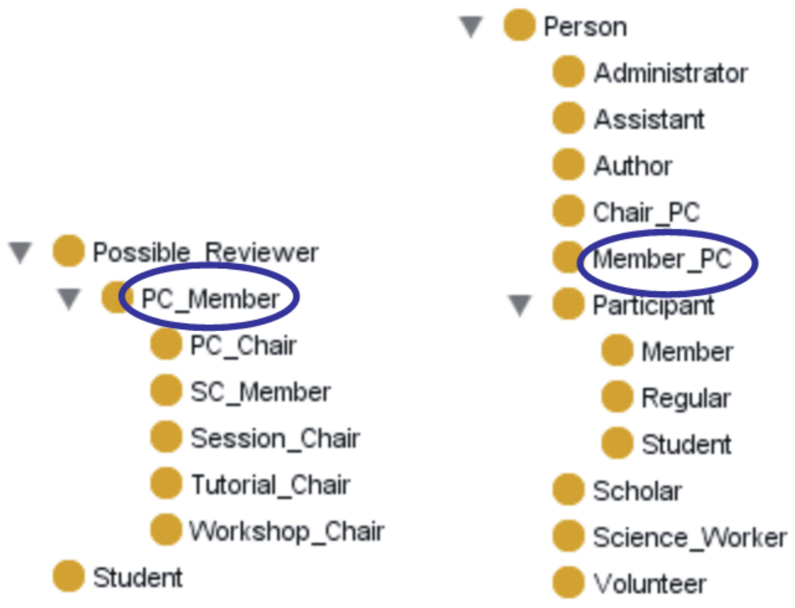


(a) IASTED (b) SIGKDD

Figure 36.1: Subtrees rooted at the concepts *human* and *person*.

As we can see, the notion of a person in SIGKDD also contains subclasses not subsumed under human in IASTED (e.g., speakers). As it is clear, however that both ontologies cover the same domain, it was decided that in this case the two concepts actually have the same intended meaning even though they do not share all subclasses.

PC_Member vs. member_PC: The concepts *PC_member* and *member_PC* are another example of correspondences that seem to be trivially correct at first sight. In this case the question is whether the ontologies assume the same set of people to belong to the program committee. A look at the hierarchies reveals that the two ontologies use a different interpretation of the set of people belonging to the PC. In particular in one case the *PC_chair* is assumed to be a member of the committee, in the other case not, see Figure 36.2. This seems to imply that the notion of *PC_member* in EKAW is more general than that in *ConfTool*. However, this is only the case if we assume that the concepts *Chair_PC* und *PC_Chair* are equivalent. Another possible interpretation is that the concepts *PC_member* and *member_PC* are equivalent but *Chair_PC* and *PC_Chair* are different concepts, namely one denoting PC chairs that are members of the PC and the other denoting PC chairs that are not members of the PC. While both interpretations are possible, the majority of workshop participants favored the first interpretation where PC chairs are the same concepts.



(a) EKAW (b) ConfTool

Figure 36.2: Subtrees containing the concepts *PC_Member* and *Member_PC*.

36.4.2 Lessons learned

The discussions at the consensus workshop revealed a number of insights about the nature of ontology matching and limitations of existing systems that provide valuable input for the design of matching tools. In the following we summarize the three most important insights gained.

Relevance of context. Probably the most important insight of the consensus workshop was that in many cases it is not enough to look at the concept names to decide whether a correspondence is correct or not. In all of the examples above, the position of the concept in the hierarchy and in some cases also the scope of the complete ontology had to be taken into account. In some cases, a decision actually requires deep ontological arguments, for instance, to distinguish between a recommendation and the actual decision made on the basis of this recommendation. For existing matching tools this means that the use of lexical matching techniques and often even of local structure matching is not sufficient. Matchers rather have to take the complete ontology and its semantics or even background knowledge about basic ontological distinctions into account. This observation is also supported by the results of the reasoning-based do2002a where automatically created correspondences often turned out to cause inconsistencies in the ontologies.

Semantic relations. All of the systems participating in the do2002a were restricted to detecting equivalences between concepts or relations respectively. It turned out that this restriction is a frequent source of errors. Often ontologies contain concepts that are closely related but not exactly the same. In many cases one concept is actually a subclass of the other. Heuristics-based matching tools will often claim these concepts to be equivalent, because they have similar features and similar positions in the hierarchy. As a result, such correspondences often become inconsistent.

We believe that matching tools that are capable of computing subsumption rather than equivalence relations are able to produce more correct and suitable correspondences.

Alternative interpretations. The example of *PC_member* illustrates the fundamental dilemma of ontology matching, which tries to determine the intended meaning of concepts based on a necessarily incomplete specification. As a result, it is actually not always possible to really decide whether a correspondence is correct or not. All we can do is to argue that a correspondence is consistent with specifications in the ontologies and with the other correspondences. In the example this leads to a situation where we actually have two possible interpretations each of which makes a different set of correspondences correct. It is not completely clear how this dilemma can be handled by matching tools. The only recommendation we can give is in favor of using methods for checking the consistency of correspondences as an indicator whether the correspondence encodes a coherent view on the system.

36.5 Evaluation via pattern-aware data mining

36.5.1 Matching patterns

Before discussing the matching patterns, it is useful to briefly consider the notion of patterns as typically treated in ontological engineering research. We will consider three categories of patterns: content patterns, logical patterns and frequent errors. *Content patterns* [Gangemi, 2005] use specific non-logical vocabulary and describe a recurring, often domain-independent state of affairs. An example is the “Descriptions&Situations” pattern, which reflects the typical way a situation (with various entities and events involved) is described using some representation. *Logical patterns*, in turn, capture the typical ways certain modeling problems can be tackled in a specific ontological language. An example is the “Classes as Property Values” pattern[2], which defines multiple ways to satisfy the need for using a class in place of a value of an OWL property. Finally, *frequent errors* (though not usually denoted as patterns, they are clearly so) describe inadequate constructions that are often used by inexperienced modelers [Rector *et al.*, 2004]. All three mentioned types of patterns are used to describe modeling behaviors that are considered as either desirable (content and logical patterns) or undesirable (frequent errors). They can be qualified as *design* patterns; indeed, ontology building is essentially an activity carried out by human intellect (at least at the level of defining logical axioms, which are hard to obtain via automated ontology learning). In contrast, *matching patterns* that will be discussed further are by themselves neither desirable nor undesirable; their desirability depends on the correctness of the correspondences. They don’t result from a deliberate activity by humans but can be detected in data output by automated matching systems.

As opposed to ontology design patterns, which concern one ontology, matching patterns deal with (at least) two ontologies. These patterns reflect the *structure of ontologies* on the one side, and on the other side they include *correspondences* between elements of ontologies. A matching pattern is a graph structure, where nodes are classes, properties or instances. Edges represent correspondences, relations between elements (e.g., domain and range of properties) or structural relations between classes (e.g., subclasses or siblings).

[2] `http://www.w3.org/TR/swbp-classes-as-values/`

The simplest (trivial) matching pattern we do not consider here only contains one element from each of the two ontologies (let us call them O1 and O2), and a correspondence between them. In our data mining experiments (described later) we employed three slightly more complex matching patterns.

The first one is depicted in Figure 36.3. The left-hand side (class A) is from O1 and the right-hand side (class B and its subclass C) is from O2. There is a correspondence between A and B and at the same time between A and C.

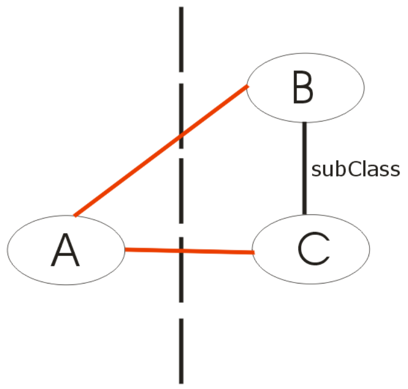


Figure 36.3: Pattern 1 – 'Parent-child triangle'.

The second pattern is depicted in Figure 36.4. It is quite similar to the previous one, but now we consider a child and a parent from each ontology and simultaneous correspondences between parents and between children.

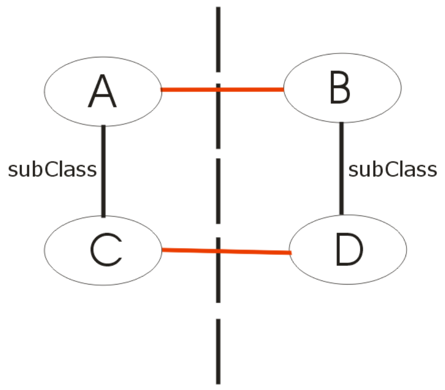


Figure 36.4: Pattern 2 – 'Matching along taxonomy'.

The third matching pattern we consider is depicted in Figure 36.5. It consists of simultaneous correspondences between class A from ontology O1 and two sibling classes C and D from ontology O2.

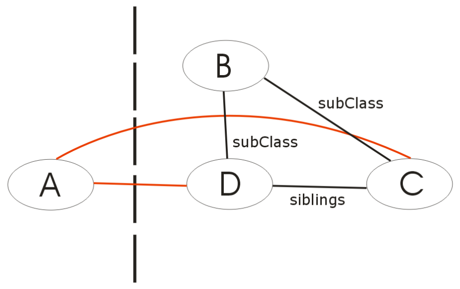


Figure 36.5: Pattern 3 – 'Sibling-sibling triangle'.

36.5.2 *4ft-Miner* overview

The *4ft-Miner* procedure is the most frequently used procedure of the *LISp-Miner* data mining system [Rauch and Šimůnek, 2005]. *4ft-Miner* mines for association rules of the form $\varphi \approx \psi/\xi$, where φ, ψ and ξ are called *antecedent*, *succedent* and *condition*, respectively. Antecedent and succedent are conjunctions of *literals*. Literals are derived from attributes, i.e., fields of the underlying data matrix; unlike most propositional mining system, they can be (at run time) equipped with complex *coefficients*, i.e., value ranges. The association rule $\varphi \approx \psi/\xi$ means that on the subset of data defined by ξ, φ and ψ are associated in the way defined by the symbol $\approx$. The symbol $\approx$, called *4ft-quantifier*, corresponds to some statistical or heuristic test over the four-fold contingency table of φ and ψ.

The task definition language of *4ft-Miner* is quite rich, and its description goes beyond the scope of this chapter. Let us only declare its two important features for our mining task: it is possible to formulate a wide range of so-called *analytic questions*, from very specific to very generic ones, and the underlying data mining algorithm is very fast due to highly optimized bit-string processing [Rauch and Šimůnek, 2005].

36.5.3 Using *4ft-Miner* for mining over matching results

For the purpose of data mining, a data matrix with each record capturing all information about one (occurrence of) correspondence was built[3]. This elementary information amounted to: name of matching *system* that detected this (occurrence of) correspondence; *validity* assigned to the correspondence by the system; types of *ontologies* ('tool', 'insider', 'web') on both sides of the correspondence; correctness *label* manually assigned to the correspondence (§36.2). In addition, there is information about *patterns* (those from §36.5) in which the given correspondence participates. There are two data fields for each of the three patterns; the first one contains the correctness *label* of the *other* correspondences within the pattern (note that there are exactly two correspondences in each of these simple patterns), and the second one contains the *validity* assigned to this *other* correspondence by the system.

The analytic questions (i.e., task settings) we formulated for *4FT-Miner* were, for example, as follows:

1. Which systems give higher/lower validity than others to the correspondences that are deemed in/correct?

[3]In total, there are 5238 records.

2. Which systems produce certain matching patterns more often than others?
3. Which systems are more successful on certain types of ontologies?

Due to limited space we do not list complete nor detailed results of the data mining process. We only present some interesting association hypotheses discovered.

For the first question, we found, for example, the following hypotheses:

– Correspondences output by Falcon with medium validity (between 0.5 and 0.8) are almost twice more often incorrect than such correspondences output by all systems (on average).
– Correspondences output by RiMOM and by HMatch with high validity (between 0.8 and 1.0) are more correct than such correspondences output by all systems (on average).

For the second question, we found, for example, the following hypotheses:

– Correspondences output by HMatch with medium validity (between 0.5 and 0.8) are more likely to connect a child with a class that is also connected (with high validity) with a parent (Pattern 1) than such correspondences with all validity values (on average).

– Correspondences output by RiMOM with high validity (between 0.8 and 1.0) are more likely to connect class C with class D whose parent B is connected (with high validity) with A, which is parent of C (Pattern 2), than such correspondences with all validity values (on average).

These two hypotheses seem to have a natural interpretation (at the level of patterns, perhaps it is not so at the level of matching systems). Pattern 1 represents a potential matching conflict, i.e., increasing the validity of one may lead to decreasing the validity of the other. On the other hand, Pattern 2 seems to evoke positive feedback between the two correspondences.

A feature of the *OntoFarm* collection that was clearly beneficial for the data mining approach to matching do2002a was the fact that it contains (far) more than two ontologies that can be matched. Thanks to that, matching patterns frequently arising because of the specific nature of some ontologies could be separated from matching patterns that are frequent in general.

36.6 Discussion

To our knowledge, there has been no systematic effort in posterior analysis of ontology alignments without reference alignment involving multiple methods like discussed in this chapter. There are only projects with which we share some isolated aspects. For example, matching patterns are implicitly considered in [Ghidini and Serafini, 2006]. However, that work focuses on heterogeneous correspondences (e.g., class to property) as a special kind of pattern. We also considered this, but it appeared too infrequently (essentially, it was only output by the COMA++ system) to allow for meaningful data mining.

The purpose of the current study was to examine multiple methods of posterior do2002a of ontology alignments, focusing on the situation when there is no reference alignment available and/or we want to obtain deeper insight into the nature of correspondences. Our results could have at least two potential uses: to give the authors of individual matching systems feedback on strong and weak points of the systems (going far beyond the usual precision/recall statistics), and to contribute to better insight of the whole research community into possible argumentation used in the ontology matching process.

Although the methods are largely different, they have certain dependencies. In particular, initial manual empirical do2002a is pre-requisite for selecting representative cases for the consensus building workshop (this role was also played by automated reasoning) as well as for subsequent data mining. Consensus workshop, in turn, helped refine the nature of matching patterns. An outline of general methodology could easily be worked out from these dependencies.

Chapter 37

Conclusions

We summarize the major findings of this deliverable as well as outline directions for future activities along its two themes, namely: (i) semantic precision and recall discussed in Chapter 23, and (ii) the results of the OAEI-2006 campaign presented in the rest of the deliverable.

For what concerns the first theme, we plan to implement these measures (semantic precision and recall) and apply them to larger sets of data (results of the OAEI[1] do2002a campaigns for instance). This requires the use of a correct and complete prover for the considered ontology languages.

For what concerns the OAEI-2006 campaign, we have several observations. Firstly, the tests that have been run in 2006 were even more complete than those of the previous years. However, more teams participated and the results tend to be better. This shows that, as expected, the field of ontology matching is becoming stronger (and we conjecture that do2002a has been contributing to this progress). Finally, the Ontology Alignment Evaluation Initiative will continue these tests by improving both test cases and testing methodology for being more accurate.

37.1 Lesson learned

From OAEI-2005 lesson learned, we have applied those concerning character encoding, new do2002a measures and having a progressive test suite in the directory case. However, we must admit that not all of them have been applied, partly due to lack of time. So we reiterate those lessons that still apply with new ones, including:

A) It is now a general trend that tools for the semantic web are more robust and compliant. As a consequence, we had comments on the tests this year that concerned problems not discovered in previous years. Obviously the tools can now better handle the ontologies proposed in the tests and they return results that are more easy to handle for the do2002a. Moreover, we had more participants able to handle large scale test cases.

B) Not all the systems from the last year campaign participated in the campaign of this year. Fortunately, the best system participated this year as well. It will be useful to investigate if this is a definitive trend, whether we are evaluating research prototypes or “serious” systems.

[1]`http://oaei.ontologymatching.org`

C) The benchmark test case is not discriminant enough between systems. It is still useful for evaluating the strengths and weaknesses of algorithms but does not appear to be sufficient anymore for comparing algorithms (that have already participated in the campaigns). We will have to look into better alternatives.

D) We have had more proposals for test cases this year (we had actively looked for them). However, the difficult lesson is that proposing a test case is not enough, there is a lot of remaining work in preparing the do2002a, e.g., reference alignments acquisition. For example, as showed in [Zhang and Bodenreider, 2007b] the absence of a reference alignment cannot be adequately compensated by the use of cross-validation. A cursory review also leaves many open questions, while establishing manually a reference alignment would require the collaboration of domain experts and adequate funding. We expect that with tool improvements, it will be easier to perform the do2002a.

E) It would be interesting and certainly more realistic, to provide some random gradual degradation of the benchmark tests (5% 10% 20% 40% 60% 100% random change) instead of a general discarding of a feature. This has not been done so far due to lack of time.

F) Last but not least, similarly to the last year the time line for this do2002a was far from ideal both from the participants and the evaluators points of view. More time must be allocated to the next campaigns.

37.2 Future plans

In the short term, future plans for the Ontology Alignment Evaluation Initiative are certainly to go ahead and to improve it. In particular, OAEI-2007 campaign[2] will be held in conjunction with the Ontology Matching workshop[3] at the annual International Semantic Web Conference in Busan, Korea. Further improvements along the technical lines include:

– Finding new real world test cases;
– Improving the tests along the lesson learned;
– Accepting continuous submissions (through validation of the results);
– Improving the measures to go beyond precision and recall (we have done this for generalized precision and recall as well as for using precision/recall graphs, and will continue with other measures);
– Drawing lessons from the new test cases and establishing general rules for consensus reference alignment acquisition and application-oriented do2002a. In particular, for the next campaign, tests were delivered earlier and we will have more time for evaluating the results.

These are only indicative lines of improvement, which are to be further refined during the forthcoming campaigns.

In the medium to long term, it is planned to continue running the Ontology Alignment Evaluation Initiative campaign as long as it is useful. At the moment the effort is tremendously useful and this is why we have increasing numbers of test cases and participants. Moreover, organisers are committed to continue running this event. While the task could be heavy, it is split among

[2]http://oaei.ontologymatching.org/2007/
[3]http://om2007.ontologymatching.org/

several organisers: this make it more acceptable. The OAEI has been created three years ago as a sustainable instrument from scratch. It is an independent initiative supported by solid institutions like INRIA or the university of Trento. It has a visible web site, steering committees and dedicated leaders. It does not require much resource to be run so there is no reason to make more elaborate plans for sustainability: as soon as this effort is useful, it will exist in one form or another, and if it becomes useless, then it would certainly have met its goal.

Chapter 38

Towards matchers recommendations

In the ontology matching field, there is no overarching matching algorithm for ontologies that is capable of serving all (heterogeneous) ontological sources. Most of the research in this area proposes new approaches based on different principles and relies on various features. These new approaches only solve small parts of "global" problems in the matching field or fill some open matching gaps [Frst and Trichet, 2005]. Therefore in general, when implementing an application using a matching approach, the corresponding algorithm is typically built from scratch and only small marginal attempt to reuse existing methods is made.

Despite an impressive number of research initiatives in the matching field, current matching approaches still feature major limitations. For example, the majority of existing approaches to ontology matching are (implicitly) restricted to processing particular classes of ontologies and thus they are unable to guarantee a predictable quality of results on arbitrary inputs. What is required are appropriate ontology matching techniques capable of coping with different levels of detail in concept descriptions [Castano *et al.*, 2004].

Hence, finding the most suited matching system for a particular application is a difficult task because there are so many different systems and so many different application characteristics. This double variability can be seen positively or negatively depending on one's capacity to take advantage of it.

The goal of this deliverable is to provide elements for choosing a matching system depending on the application to be developed. Most of these elements are of generic nature: we describe methodologies for evaluating the adequacy between matchers and applications. We also apply these methodologies to actual matching systems and a use case taken from Knowledge web use cases [Léger *et al.*, 2005].

The methodology followed in this deliverable is summarised by Figure 38.1. Our goal is to define matching system profiles (Chapter 42) and application profiles (Chapter 39) and to base recommendations on the matching of these two types of profiles (Chapter 44). For obtaining the matcher profiles we ground our work on:

– identifying what are the characteristics of these matchers as well as the needs for the applications (Chapter 40);
– designing strategies for obtaining the actual profiles, for instance through do2002a (Chapter 41);
– extracting matcher profiles through either literature review, developer survey or do2002a results (Chapter 42).

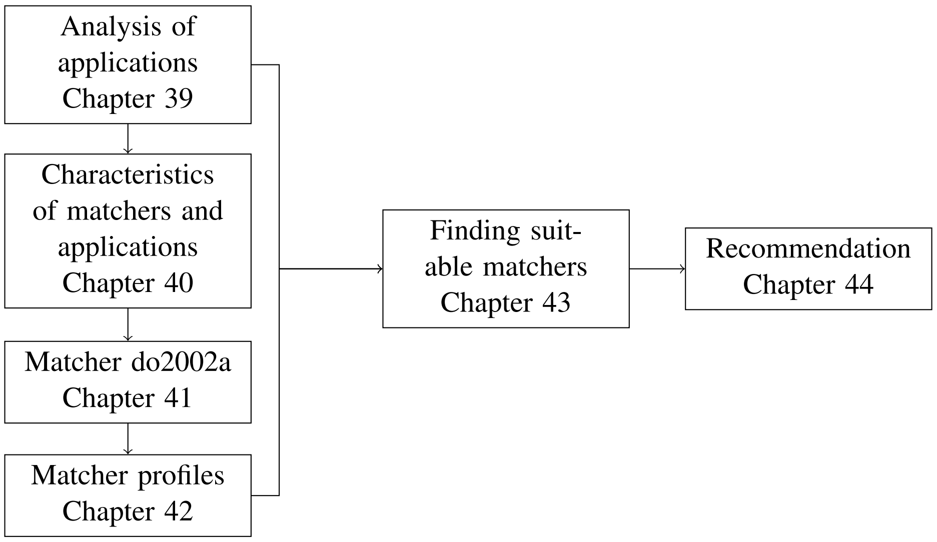

Analysis of applications Chapter 39
Characteristics of matchers and applications Chapter 40
Matcher do2002a Chapter 41
Matcher profiles Chapter 42
Finding suitable matchers Chapter 43
Recommendation Chapter 44

Figure 38.1: The process adopted by this deliverable (as well as set of chapters).

The matcher profiles can be compared to individual and generic application profiles through multi-criteria decision methods (Chapter 43). This methodology is applied to use cases introduced in Deliverable 1.1.4 and more specifically to use case 1 (Chapter 44).

Chapter 39

Analysis of applications

We provide here a summary analysis of the requirements of applications with regard to ontology matching. We first consider a typology of applications (§39.1), examine the requirements that these applications may raise (§39.2) and apply these results to the relevant Knowledge web use cases (§39.3).

39.1 Types of applications

In previous deliverables [Euzenat *et al.*, 2004a] and [Euzenat and Shvaiko, 2007], several classes of applications are considered (they are thoroughly described in the above references, we only summarized them here). They are the following:

Ontology evolution uses matching for finding the changes that have occured between two ontology versions;

Schema integration uses matching for integrating the schemas of different databases under a single view;

Catalog integration uses matching for offering a integrated access to online catalogs;

Data integration uses matching for integrating the content of different databases under a single database;

P2P information sharing uses matching for finding the relations of ontologies used by different peers;

Web service composition uses matching between ontologies describing service interfaces in order to compose web services by connecting their interfaces;

Multi agent communication use matching for finding the relations between the ontologies used by two agents and translating the messages they exchange;

Context matching in ambient computing uses matching of application needs and context information when application and devices have been developed independently and use different ontologies;

Query answering uses ontology matching for translating user queries about the web;

Semantic web browsing uses matching for dynamically (while browsing) annotating web pages with partially overlapping ontologies.

These kinds of applications have been analysed in order to establish their requirements with regard to matching systems.

39.2 Application requirements

This analysis leads to different requirements for different applications. We summarise in Table 39.1 what we have found to be the most important requirements to matching solutions in the considered applications. These requirements concern:

– the type of available input a matching system can rely on, such as schema or instance information. There are cases when data instances are not available, for instance due to security reasons [Clifton *et al.*, 1997] or when there are no instances given beforehand. Therefore, these applications require only a matching solution able to work without instances (here schema-based method).

– some specific behaviour of matching, such as requirements of (i) being *automatic*, i.e., not relying on user feed-back; (ii) being *correct*, i.e., not delivering incorrect matches; (iii) being *complete*, i.e., delivering all the matches; and (iv) being performed at *run-time*.

– the use of the matching result as described above. In particular, how the identified alignment is going to be processed, e.g., by merging the data or conceptual models under consideration or by translating data instances among them.

The above requirements together with applications from which they come are summarised in Table 39.1. Ontology evolution is typically used at design time for transforming an existing ontology which may have instances available. It requires an accurate (i.e., correct and complete) matching, but can be performed with the help of users. Schema, Catalogue and Data integration are also performed off line but can be used for different purpose: translating data from one base to another, merging two databases or generating a mediator that will be used for answering queries. They also will be supervised by a human user and can provide instances. Other applications are rather performed at runtime. Some of these like P2P information sharing, query answering and semantic web browsing are achieved in presence of users who can support the process. They are also less demanding in terms of correctness and completeness because the user will directly sort out the results. On the other hand, web-service composition, multiagent communication and context matching in ambient computing require matching to be performed automatically without assistance of a human being. Since, the systems will use the result of matching for performing some action (mediating or translating data) which will be feed in other processes, correctness is required. Moreover, usually these applications do not have instance data available.

Some of these hard requirements can be derived into comparative (or non-functional) requirements such as *speed*, resource consumption (in particular memory requirements), degree of correctness or completeness. They are useful for comparing solutions on a scale instead of absolutely. Moreover, they allow to trade a requirement, e.g., completeness, for another more important one, e.g., speed.

Another dimension along which these applications differ is the operation for which they perform matching:

– ontology engineering requires the ability to *transform* relevant ontologies or some parts of these ontologies into an ontology focusing on a domain of interest being modeled or to generate a set of bridge axioms that will help in identifying corresponding concepts (the transformations apply at the ontological level);

Application	instances	run time	automatic	correct	complete	operation
Ontology evolution	√			√	√	transformation
Schema integration	√			√	√	merging
Catalog integration	√			√	√	data translation
Data integration	√			√	√	query mediation
P2P information sharing		√				query mediation
Web service composition		√	√	√		data mediation
Multi agent communication		√	√	√	√	data translation
Context matching in ambient computing		√	√	√		data translation
Query answering	√	√				query reformulation
Semantic web browsing	√	√		√		navigation

Table 39.1: Summary of applications requirements.

– schema integration requires the ability to *merge* the schemas under consideration into a single schema (the transformations apply at the ontological level);
– data integration requires the ability to *translate data* instances residing in multiple local schemas according to a global schema definition in order to enable query mediation over the global schema;
– peer-to-peer systems and more generally query mediation systems require bidirectional *mediators* able to translate queries (ontological level) and translate back answers (data level);
– agent communication requires *translators* for messages sent from one agent to another, which apply at the data level; similarly, semantic web services require one-way data translations for composing services.

These requirements are reported in the "operation" column of Table 39.1.

39.3 Application to use cases

The use cases that have been provided in [Léger *et al.*, 2005], are recalled in Table 39.2 and classified with regard to the kind of application they are with regard to their ontology matching needs. This classification reflects only the main category of needs for matching. Some of these application have in reality several heterogeneity problems that can be solved differently and most of them cannot be reduced to ontology matching and are far wider.

In the present deliverable, we will more closely consider the first use case in Recruitements and job finding.

39.4 Summary

We have now a first idea of the application needs in order to define more finely the characteristics of matchers in the next chapter. These are generic requirements that apply to a whole class

Use case	Company	Matching	Type
Recruitments	WoldwideJobs	$\surd$	Data integration
B2C marketplace for tourism	France Telecom	$\surd$	Query answering
News aggregation	Neofonie	$\surd$	Semantic web browsing
Product lifecycle management	Semtation		
Real estate management	Trenitalia	$\surd$	P2P information sharing
Integrated access to biological data	Robotiker	$\surd$	Data integration
Geoscience project memory	IFP		
Hospital information system	L&C Global		

Table 39.2: Identified Knowledge web use cases and their correspondences in the classification of Table 39.1

of applications and must be refined into specific requirements applying to a particular application. Specific requirements for applications will be more thoroughly defined in Chapter 40 by characterizing matcher characteristics.

These generic requirements are used in Chapter 40 to be a basic set of requirements, in Chapter 41 for designing do2002a procedures related to applications as well as in Chapter 42 to establish matcher profiles.

Chapter 40

Characteristics of matching approaches (and applications)

Having a definition of the problem to be solved and the particular requirements regarding the final application, one must decide which matching algorithms are to be applied to satisfy these specification and to obtain the desired output. This depends on the characteristics of the applications (about input, output, process, etc.) and those of the available matchers. Possible attributes, that could have an impact on the selection of an adequate matching approach, must be defined in order to resolve this issue. Accounting for the empirical findings of different case studies in ontology engineering [Mochol and Paslaru Bontas Simperl, 2006; Paslaru Bontas and Mochol, 2005; Paslaru Bontas *et al.*, 2005], and regarding the requirements (cf. Chapter 39) collected during the development of different Semantic Web application scenarios [Bizer *et al.*, 2005; Garbers *et al.*, 2006][1], as well as during the intensive collaborations with ontology and software engineers, six groups of factors, called dimensions, have been defined as relevant for the matching selection process.

These dimensions are the main aspects that must be taken into account during the examination of the suitability of a single matching approach for the solving of a given problem: (i) *input characteristics* that takes into account the ontologies to be matched; (ii) *approach characteristics* describes the matching algorithms themselves; (iii) *output characteristics* defines the desired result of the matching execution; (iv) *usage characteristics* takes into account the different situations where the approaches have been used; (v) *documentation characteristics* points out the existence and type of the documentation; and (vi) *cost characteristics* addresses the costs which have to be paid for the usage of the algorithm.

The dimensions constitute the superficial collection for matcher attributes: they are the first level of the *multilevel characteristics for matching approaches*. The multilevel characteristics is organized in the form of a taxonomy where dimensions are defined by sets of factors and these are described by the attributes. These characteristics can be illustrated as a taxonomy (cf. Figure 40.1) where the child nodes describe and represent the parent nodes' properties [Lozano Tello and Gomez Perez, 2004].

In the following sections we briefly describe some of the factors of each dimension and state others in the form of tables (cf. Table 40.1, 40.2, 40.3, 40.4, 40.5, 40.6). The content of the tables are arguable and could have been presented in another way. However, these tables are the basis

[1]Projects: (i) Wissensnetze,[2], (ii) Reisewissen, [3], (iii) SWPatho, [4], (iv) Knowledge Web [5]

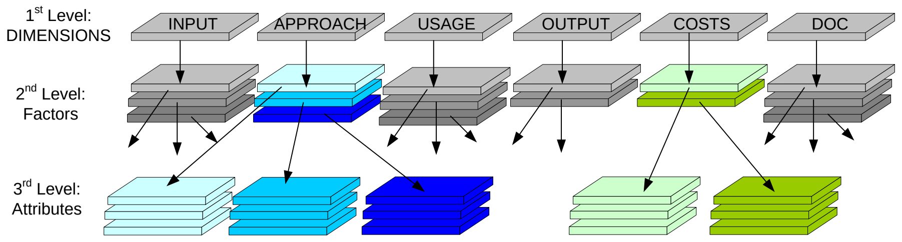


Figure 40.1: Multilevel characteristics with dimensions, factors and attributes

for the questionnaire that is used in §43.3. The questionnaire offered more details about what was expected than these table content. We did not tried to improve a posteriori the tables, but rather presented them as such. The questionnaires may be improved in the future.

40.1 Input characteristics

The first step towards the analysis of the matching characteristics is the examination of the matching input. In our opinion, the attributes that describe the input are the most important and relevant criteria that play a crucial role in the selection of the appropriate algorithm. Despite the relatively large number of promising matching approaches their limitations with respect to certain ontology characteristics have often been emphasized in recent literature [Giunchiglia and Shvaiko, 2003b; Madhavan *et al.*, 2001; Melnik *et al.*, 2002; Shvaiko, 2006; Shvaiko and Euzenat, 2005]. The *input characteristic* dimension describes not only the heterogeneity of the sources that are to be matched, e.g. *size* (some matchers perform well on relatively small inputs), *natural language* used for the definition of concepts (some algorithms require certain natural language) and *input structure* (some matchers do not perform well on heterogeneous structures [Giunchiglia and Shvaiko, 2003b]), but also takes into account *external sources*, which a matching algorithm can use for its execution (cf. Table 40.1).

Attribute	**Description**
Factor: Input Size (algorithm capable of handling:)	
number of ontologies	number of different ontologies to be matched (two or more)
size of input	number of ontological primitives (concepts, properties, axioms) to be matched: *small* (up to 100 primitives), *middle* (101 - 500 primitives), *large* (501 - 1,000 primitives), *extra large* (over 1,000 primitives)
size of instances	number of instances to be matched: *no instances*, *small* (up to 500 instances), *medium* (501 - 1,000 instances), *large* (1,001 - 10,000 instances), *extra large* (over 10,000 instances)
number of concepts	number of concepts to be matched: *small* (up to 100 concepts), *medium* (100 - 500 concepts), *large* (500 - 1,000 concepts), *extra large* (over 1.000 concepts)
number of relations	number of relations to be matched: *small* (up to 30 relations), *medium* (31 - 100 relations), *large* (100 - 1,000 relations), *extra large* (over 1,000 relations)
number of axioms	number of axioms to be matched: *no axioms*, *small* (up to 30 axioms), *medium* (31 - 100 axioms), *large* (100 - 1,000 axioms), *large* (over 1,000 axioms)
Factor: Input category (algorithm capable of handling:)	
glossary	a list of terms with their definitions
thesaurus	a list of important terms (single-word or multi-word) in a given domain and a set of related terms for each term in the list
taxonomy	indicates only class/subclass relationship (hierarchy) [Dogac *et al.*, 2002]
to be continued ...	

Attribute	**Description**
DBschema	often does not provide explicit semantics for their data
ontology	an explicit specification of a conceptual [Gruber, 1995]; describes a domain completely [Dogac *et al.*, 2002]
Factor: Input formality level [Uschold and Jasper, 1999] (algorithm capable of handling:)	
(highly/semi) informal ontology	expressed loosely in natural language or in a restricted and structured form of natural language
semi-formal ontology	expressed in an artificial, formally defined language
(rigorously) formal ontology	meticulously defined terms with formal semantics, theorems and proofs of such properties as soundness and completeness
Factor: Input model type [Guarino, 1998] (algorithm capable of handling:)	
task ontology	model build for a generic task (e.g. diagnosing)
domain ontology	model of a generic domain (e.g. medicine) or part of the world
application ontology	model built for a specific application; concepts depend on both a particular domain and task, which are often specializations of both the related ontologies
upper-level ontology	model of the common objects that are generally applicable across a wide range of domain ontologies; it describes very general concepts (e.g. space, time)
Factor:Input type (algorithm capable of handling:)	
scheme	schema-based matcher
instance	instance/content-based matchers
Factor: External sources (algorithm is able to handle /to provide:)	
additional user input	most matchers rely not only on the input to be matched (like schemas or instances) but also on auxiliary information
previous matching decision	
training matches	
domain specific resources/ constrains	
domain constrains	
list of valid domain values	
dictionary	
missmatch information	
matching rules	
global schemas	
Factor: Input natural language (NL) (algorithm is:)	
NL-specific (one language)	the approach is dependent on one natural language
NL-specific (many languages)	the approach is dependent on more than one natural languages
NL-independent	the approach is language independent
Factor: Input representation language (RL) [Uschold and Jasper, 1999] (algorithm is:)	
RL-specific (one language)	the approach is dependent on one rep. language
RL-specific (many languages)	the approach is dependent on more then one rep. languages
RL-independent	the approach is independent on rep. language
Factor: Input structure (algorithm capable of handling:)	
tree structure	the approach can handle only tree-structures
DAGs structure	the approach can handle directed acyclic graphs structures
graph structure	the approach can handle (heterogenous) graph structures
is-a relations	the approach can handle is-a relations
heterogeneous relations	the approach can perform additionally to the "is-a relations" also on other relations

Table 40.1: Input characteristics

40.2 Approach characteristics

The second crucial dimension characterizes the matching approaches themselves. The corresponding factors and attributes compile a list of matcher features that are empirically proved to have an impact on the quality of matching tasks. They consider e.g. the common classification of the approaches [Do *et al.*, 2002; Rahm and Bernstein, 2001; Shvaiko, 2006] and distinguish between *individual algorithms* [Giunchiglia and Shvaiko, 2003b; Stumme and Mädche, 2001] and combinations of the individual algorithms: *hybrid* and *composite solutions*. A hybrid approach [Madhavan *et al.*, 2001] follows a *black box paradigm*, in which various individual matchers are synthesized into a new algorithm, while the composite matchers allow an increased user interaction [Do and Rahm, 2002; Doan *et al.*, 2004]. The *approach characteristics* also takes into account issues like processing type, matching ground and execution parameter (cf. Table 40.2).

Attribute	**Description**
Factor: Matcher Type (algorithm is a(n):	
individual matcher	computes a mapping based on a single matching criteria
combined matcher	uses multiple individual matchers
Factor: Processing (algorithm supports:)	
manual execution	manual execution
gray box paradigm	semi-automatic execution where the human intervention is possible
black box paradigm	automatic execution without human intervention
manual preprocessing allowed/required	human intervention before execution is allowed or even required
manual postprocessing allowed/required	human intervention after execution is allowed or even required
Factor: Execution Type (algorithm supports:)	
simultaneous execution	the single matching algorithms (within a composite matcher) can be executed simultaneously
sequential execution	the single matching algorithms (within a composite matcher) can be executed sequentially
Factor: Kind of Similarity Relation (algorithm performs:)	
syntactic matching	similarity based on syntax driven techniques and syntactic similarity measures; relation computed between labels at nodes [Shvaiko, 2006]
semantic matching	relation computed between concepts at nodes [Shvaiko, 2006]
Factor: Matcher Level (algorithm can perform on:)	
element level	match performed for individual schema elements
structure level	match performed for complex schema structures
atomic level	elements at the finest level of granularity are considered e.g. attributes in an XML schema [Rahm and Bernstein, 2001]
non-atomic (higher) level	e.g. XML elements
Factor: Matching Ground	
heuristic	“guessing” relations between similar labels or graph structures [Shvaiko, 2004a]
formal	uses formal techniques (e.g. can have model-theoretic semantics which is used to justify the results) [Shvaiko, 2004a]
Factor: Semantic Codification Type(algorithm uses:)	
implicit techniques	syntax driven techniques [Shvaiko, 2004a](e.g. considers labels as strings)
explicit techniques	exploit the semantics of labels [Shvaiko, 2004a]; uses an external sources for assessing the meaning of labels
Factor: Execution Parameter (algorithm needs:)	
max time of execution	describes the maximal time needed for execution
max disc space for execution	describes the maximal disc space needed
precision	expresses the proportion of relevant retrieved matches [van Rijsbergen, 1975]
recall	expresses the proportion of relevant documents retrieved [van Rijsbergen, 1975]

Table 40.2: Approach characteristics

40.3 Usage characteristics

One of the fundamental requirements for the realization of the vision of the fully developed Semantic Web are proven ontology matching algorithms. Though containing valuable ideas and techniques some of the current matching approaches lack exhaustive testing in real world scenarios. Considering this problem and additionally making allowance for the fact that some of the algorithms cannot be applied across various domains to the same effect [Giunchiglia and Shvaiko, 2003b], it is important to know, if a particular approach has already been successfully adapted for different *domains*, *applications* and *tasks*. Additionally, the *usage characteristics* dimension also considers *different types of users*: ontology engineers who e.g. look for means to compare sources for building a new ontology or Web Services seeking automatized methods to generate mediation ontologies (cf. Table 40.3).

Attribute	**Description**
Factor: Usage goal (algorithm is built for:)	
local use	the matcher is run on the local machine
network use	the matcher is accessible on a network
internet-based use	the matcher service is available through internet
Factor: Application Area (algorithm is built for:)	
reuse of sources	the matching approach is deployed for ontology reuse which may be defined as a process in which available knowledge is used to generate new ontologies
usage of sources	the matching approach is applied for use ontologies (within an application) e.g. to compare profiles
integration	reuse of available source ontologies within a range in order to build a new ontology which serves at a higher level in the application than that of various ontologies in ontology libraries [Li *et al.*, 2005]
transformation	associated with the ontology evolution that uses matching for finding the changes that have occurred between two ontology versions
merging	the matching approach is used for integrating the schemas of different databases under a single view
translation	ontology translation is required to translate data sets, generate ontology extensions and query through different ontologies [Dou *et al.*, 2003] e.g. (i) catalog integration - matchers are used to offer a integrated access to online catalogs, (ii) multi agent communication - matchers are used to find the relations between the ontologies used by two agents and translating the messages they exchange, (iii) context matching in ambient computing uses matching approaches of application needs and context information when application and devices have been developed independently and use different ontologies
query answering	can be applied to data integration or P2P information sharing where matching approaches are used to find the relations of ontologies used by different peers
data mediation	applied within web service composition in which matchers are used between ontologies to describe service interfaces in order to compose web services by connecting their interfaces;
query reformulation	uses matcher to translate user queries about the web
navigation	uses matchers to annotate web pages with partially overlapping ontologies
Factor: Usage type (algorithm is:)	
human applicable	approach can be used only by humans (human interaction indispensable)
machine applicable	approach can be used by machine as a service
Factor: Adaptation ability (algorithm has been applied for:)	
number of domains	number of different domains the matching approach was applied to
number of applications	number of different applications the matching approach was applied to
number of tasks	number of different tasks the matching approach was applied to
reference of usage	the approach as has been utilized by other users

Table 40.3: Usage characteristics

40.4 Output characteristics

In addition to the input, approach and usage dimensions, the *output characteristics* (cf. Table 40.4) plays a decisive role in the process of selecting the suitable matching algorithm. Depending on the given requirements, an application may need a matcher that considers only some of elements of the schemas, while other systems may mandate a match for all elements. One of the key factors in this dimension is the *cardinality* (global vs. local cardinality) which specifies whether a matcher compares one or more elements of one scheme with one or more elements of another scheme (in some cases the results are based on a one-to-one mapping between taxonomies [Doan *et al.*, 2001] and in others on one-to-many).

Attribute	**Description**
Factor: Output type	
deliver relations	the output is not restricted to correspondence of equivalence
deliver value	e.g. matcher used to determine the semantic similarity between concepts
deliver understandable (for humans) results	matcher delivers some explanations of the results
Factor: Matching Cardinality	
global 1:1	relationship cardinalities between entities w.r.t different entities [Rahm and Bernstein, 2001] e.g. one-to-one or many-to-many alignments.
global n:1	
global 1:m	
global n:m	
local 1:1	relationship cardinalities between entities w.r.t an individual correspondence [Rahm and Bernstein, 2001] e.g., simple or complex correspondences
local n:1	
local 1:m	
local n:m	
Factor: Execution Completeness	
full match	considers all elements of the schemes
partial match	considers only some elements of the schemes
injective match	distinct elements of the domain is mapped to distinct elements of the range
surjective match	all elements of the range are mapped to elements of the domain

Table 40.4: Output characteristics

40.5 Documentation characteristics

Due to the fact that documentation is an essential part of every software product and, for engineering purposes, often more important than the program code [Humphrey, 1999] the information about its quality and clarity can be significant for the selection of an approach. Furthermore, since one of the goals of documentation is to provide sufficient information so that an architecture can be analyzed for suitability to the purpose [Clements *et al.*, 2002], it could be a determining coefficient for the selection of a particular algorithm, especially if the algorithm is to be reused in a different context from the domain or application it was originally developed for (cf. Table 40.5).

Attribute	**Description**
quality of documentation	quality of the available documentation
clarity of documentation	clarity of the available documentation
clarity of maturity description	clarity of the description of the approach's maturity
availability of examples	are examples of the approach available

Table 40.5: Documentation characteristics.

40.6 Cost characteristics

The last dimension, *cost characteristics*, describes the financial factors regarding the (commercial[6]) usage of a single matching approach like the matcher licence or the access to the appropriate matcher interface (cf. Table 40.6).

Attribute	**Description**
costs of matcher licence	the costs entailed to acquire the matcher licence
costs of matcher tool licence	the costs entailed to acquire the tools matcher have been developed with
costs of access matcher interface	the costs entailed to acquire the use of the interface

Table 40.6: Cost characteristics.

40.7 Summary

We have investigated the various dimensions of matchers and the characteristics of these matchers along these dimensions. This inventory is very precise and goes beyond most published characterizations of matching systems in terms of attributes [Rahm and Bernstein, 2001; Kalfoglou and Schorlemmer, 2003b; Shvaiko and Euzenat, 2005; Huza *et al.*, 2006; Euzenat and Shvaiko, 2007]. Its goal is to be able to establish more precisely the adequacy of a matcher with an application. It will be used, in particular for gathering information about available matching systems and choosing one of these (see Chapter 42.2 and 43.3).

[6]At the moment there is no commercial automatic matching tool. Nevertheless to be able to deal with such problem in the future we are taking already now these criteria into account.

Chapter 41

Benchmarking matching systems

Once we have determined the characteristics on which matchers can be compared, it is necessary to assess the value of these characteristics applying to matchers. This can be achieved through literature review, direct survey of developers or evaluating matching systems. We present here two ways to evaluate matching systems. The first way is benchmarking, i.e., comparing systems on a range of tests that allows to assess the behaviour of systems in a number of well-characterized conditions. Another way consists of designing an do2002a procedure which is fully related to the application to be developed.

We provide an extensive presentation of benchmarking through the first three sections and a possible specific application-specific do2002a related to Knowledge web use case 1.

41.1 Principles

In order to evaluate the capability of matching systems to really work with some kind of input, we have developed a complete set of benchmarks which goal is to finely test the results of these systems in well identified situations.

These tests have proved useful at the beginning of the Ontology Alignment Evaluation Initiative and their first version had been presented in [Euzenat *et al.*, 2004b]. We describe here the extended version that has been used for two years. Their results in finding the adequacy of a matching system for an application is given in Section 42.3.

The tests consist of choosing some ontology and systematically generating a new ontology by discarding some information from it, e.g., names, properties, subclass relation. A reference alignment between the two ontologies can be generated in a similar way since what corresponds to what is clear. Matching system must provide an alignment between these two ontologies tht will be compared with the reference alignment. This is useful to evaluate how the system behave when this information is lacking.

The domain of this reference ontology is Bibliographic references. It is, of course, based on a subjective view of what must be a bibliographic ontology. There can be many different classifications of publications (based on area, quality, etc.). We choose the one common among scholars based on mean of publications; as many ontologies below (tests #301-304), it is reminiscent to BibTeX.

The complete ontology is that of test #101. This reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals.

It is based on the one of the first EON Ontology Alignment Contest benchmark which has been improved by comprising a number of circular relations that were missing from the first test. This set is also more complete (the 2004 benchmark only had the 16 easier ontology pairs). Test numbering (almost) fully preserves the numbering of the first EON contest so that results can be compared.

The reference ontology is put in the context of the semantic web by using other external resources for expressing non bibliographic information. It takes advantage of FOAF[1] and iCalendar[2] for expressing the People, Organization and Event concepts. Here are the external reference used:

http://www.w3.org/2002/12/cal/#:Vevent (defined in http://www.w3.org/2002/12/cal/ical.n3 and supposedly in http://www.w3.org/2002/12/cal/ical.rdf).

http://xmlns.com/foaf/0.1/#:Person (defined in http://xmlns.com/foaf/0.1/index.rdf)

http://xmlns.com/foaf/0.1/#:Organization (defined in http://xmlns.com/foaf/0.1/index.rdf).

This reference ontology is a bit limited in the sense that it does not contain subclasses of several non comparable classes. Similarly the kind of proposed alignments is still limited: they only match named classes and properties, they mostly use the ”=” relation with confidence of 1. The ontologies are described in OWL-DL and serialized in the RDF/XML format.

This data set can be considered as correct by construction. It is not realistic nor very hard: it is based on small tests and offers some easy ways to reach the correct result. It is especially made for evaluating the strengths and weaknesses of the matching systems.

41.2 The test set

Table 41.1 summarizes what has been retracted from the reference ontology. There are here 6 categories of alteration:

Entity labels Name of entities that can be replaced by (R/N) random strings, (S)ynonyms, name with different (C)onventions, (F) strings in another language than English.

Comments Comments can be (N) suppressed or (F) translated in another language.

Specialization Hierarchy can be (N) suppressed, (E)xpanded or (F)lattened.

Instances can be (N) suppressed

Properties can be (N) suppressed or (R) having the restrictions on classes discarded.

Class composition can be (E)xpanded, i.e., replaced by several classes or (F)latened.

The alteration results in a set of 46 pairs of ontologies (out of $2^6 = 64$ ontology pairs that could be obtained by all the Boolean combinations of alterations or the full set of $5 \times 3 \times 4 \times 2 \times 3 \times 3 = 1080$ combinations). This set is more restricted because some of the elaborate modifications have only been applied once. This ensures a good coverage of all the situations in which an alignment algorithm can find itself and provides a good view of strengths and weaknesses of algorithms.

[1]http://xmlns.com/foaf/0.1/

[2]http://www.w3.org/2002/12/cal/

Test	Entity labels	Comments	Subsumption hierarchy	Instances	Properties	Class composition	Comment	Used	2004	Aggregate
101							Reference alignment	√	√	liph
102							Irrelevant ontology		√	
103							Language generalization	√	√	
104							Language restriction	√	√	
201	R						No names	√	√	phi
202	R	N					No names, no comments	√	√	phi
203		N					No comments (was misspelling)	√		phi
204	C						Naming conventions	√	√	phi
205	S						Synonyms	√	√	phi
206	F	F					Foreign names	√	√	phi
207	F							√		phi
208	C	N						√		phi
209	S	N						√		phi
210	F	N						√		phi
221			N				No specialisation	√	√	lip
222			F				Flatenned hierarchy	√	√	lip
223			E				Expanded hierarchy	√	√	lip
224				N			No instance	√	√	lph
225					R		No restrictions	√	√	hil
226							No datatypes			
227							Unit difference			
228					N		No properties	√	√	hil
229							Class vs instances			
230						F	Flattened classes	√	√	hil
231						E	Expanded classes	√		hil
232			N	N				√		lp
233			N		N			√		li
236				N	N			√		lh
237			F	N				√		lp
238			E	N				√		lp
239			F		N			√		li
240			E		N			√		li
241			N	N	N			√		l
243			N	N						
244			N		N					
246			F	N	N			√		l
247			E	N	N			√		l
248	N	N	N					√		ip
249	N	N		N				√		ph
250	N	N			N			√		hi
251	N	N	F					√		ip
252	N	N	E					√		ip
253	N	N	N	N				√		p
254	N	N	N		N			√		i
257	N	N		N	N			√		h
258	N	N	F	N				√		p
259	N	N	E	N				√		p
260	N	N	F		N			√		i
261	N	N	E		N			√		i
262	N	N	N	N	N			√		∅
265	N	N	F	N	N			√		∅
266	N	N	E	N	N			√		∅

Table 41.1: Table of tests and altered features (test numbers are briefly described in the corresponding section). Some of these tests where not used in 2004 and some other are still not used.

Table 41.1 present all the tests and their generation principles. In order to provide a readable do2002a in Chapter 42.3, we have grouped together several characteristics of these tests in order to evaluate the dependency of each system on particular characteristics. Entity labels and comments are considered as (l)abels, subsumption (h)ierarchy is a category alone, (i)nstances and (p)roperties and class composition are grouped together. When these features have not been altered, the cell corresponding to this characteristics is left blank. This is reflected by the aggregate column of Table 41.1.

We provide below the description of all the tests as they where used in the first experiment, this should provide a raw blueprint on what is to be expected of such a competence benchmark tests. The subsection number is the number of the test in Table 41.1.

41.2.101 Identity

This simple test consists of aligning the reference ontology with itself.

41.2.201 No names

Each label or identifier is replaced by a random one.

41.2.202 No names, no comment

Each label or identifier is replaced by a random one. Comments (rdfs:comment and dc:description) have been suppressed as well.

41.2.203 Misspelling of names

Each label or identifier is replaced by a misspelled one. Comments (rdfs:comment and dc:description) have been suppressed as well.

41.2.204 Naming conventions

Different naming conventions (Uppercasing, underscore, dash, etc.) are used for labels. Comments have been suppressed.

41.2.205 Synonyms

Labels are replaced by synonyms. Comments have been suppressed.

41.2.206 Foreign names

The complete ontology is translated to another language than English (French in the current case, but other languages would be fine).

41.2.221 No hierarchy

All subclass assertions to named classes are suppressed.

41.2.222 Flattened hierarchy

A hierarchy still exists but has been strictly reduced (by suppressing one intermediate class out of two).

41.2.223 Expanded hierarchy

Numerous intermediate classes are introduced within the hierarchy.

41.2.224 No instances

All individuals have been suppressed from the ontology.

41.2.225 No restrictions

All local restrictions on properties have been suppressed from the ontology.

41.2.226 No datatypes

In this test all datatypes are converted to xsd:string.

41.2.227 Unit differences

(Measurable) values are expressed in different datatypes.

41.2.228 No properties

Properties and relations between objects have been completely suppressed.

41.2.229 Class vs instances

Some classes have become instances.

41.2.230 Flattening entities

Some components of classes are expanded in the class structure itself (e.g., year, month, day attributes instead of date).

41.2.231 Multiplying entities

Some classes are spread over several classes. This is the opposite as the previous test: intermediate classes and objects aggregating properties are created (e.g., an inproceedings reference with booktitle and conference properties is transformed into a inproceedings that refers to a proceedings reference that refers to a conference object).

41.3 Application-specific do2002a

So far matcher do2002a has been considered in general. However, the do2002a could also be considered in the context of a particular application or a particular kind of applications. Application specific do2002a is dedicated to find a suitable system for a particular task. This is especially useful for application designers who need to integrate a matching system.

Application specific do2002a can be carried out by having a specific do2002a setting. This has the advantage of being more realistic than artificial testbenches and of providing very specific information, but the drawback to be changed for each different application.

An application specific do2002a has to start with a selection of the task corresponding to the application. It is moreover useful to set up experiments which do not stop at the delivery of alignments but carry on with the particular task. This is especially true when there is a clear measure of success of the overall task. Such a setting assists in focusing on the most useful issues for the task. For instance, it may be the case that the gain in accuracy in one algorithm over another is not useful for the task while the gain in speed of the latter really matters. If no clear measure is available, then using a weighted aggregation measure like suggested above would help.

Nevertheless, it is extremely difficult to determine the do2002a value of the matching process independently. The effects of other components of the overall application have to be carefully filtered out.

There are several problems associated with this approach:

– It will be difficult to account for the performances of matching algorithms if the systems which carry the task are different. If these system are a single system, then the do2002a would be simpler in comparing the alignments and applying a specific metric;

– Very often the matching systems are considered as semi-automatic (i.e., a user must control the result). The task cannot be accomplished in isolation (and this brings back the issue of involving the user in the do2002a loop).

– This would require a specific setting for each task.

As an example, let us consider Knowledge web use case 1 dedicated to human resources. Its main role is to match job offers to applications. The success criterion here is matching adequate applications to job offers. Deciding which application to choose is difficult but there is no difficulty in discarding non relevant applications.

In this recruitment application the data exchange between employers, job applicants and job portals is based on a set of shared vocabularies describing domain relevant terms: occupations, industrial sectors and skills. These commonly used vocabularies can be formally defined by means of a human resource ontology. The ontology represents domain specific knowledge and might be used to determine semantic similarity between data describing job applications and job offers. Furthermore, semantic matching is a matching technique that can combine annotations using controlled vocabularies with background knowledge about a certain application domain. In addition, to enable an accurate scheme validation of the incoming data the human resource ontology is modelled using OWL which contains over 8.000 concepts and less than 100 different properties[3]. The descriptions of job postings and applicants' profiles (instance data) are stored in RDF using the vocabulary defined by the human resource ontology which does not contain any axioms.

[3]These ontologies are not freely accessible and cannot be presented in more depth.

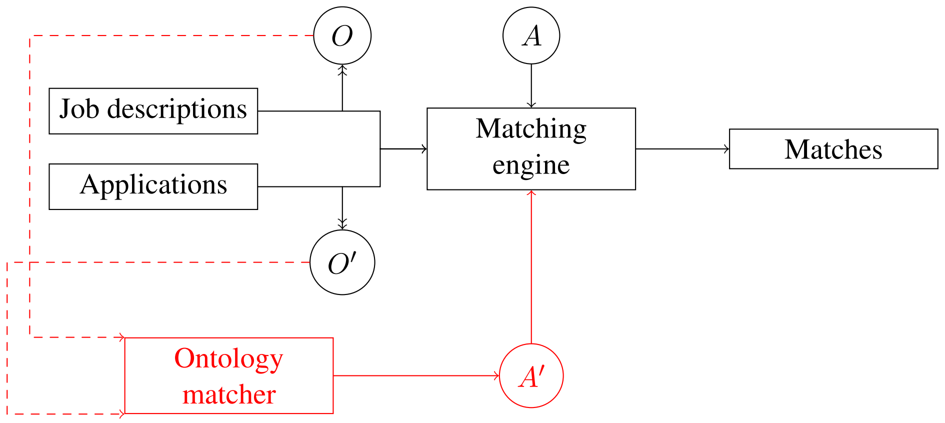


Figure 41.1: The job-application matching cases. The existing application (in black) match job descriptions to applications by using a wired ontology alignment (A). The modified application (in red) uses ontology matching for generating an alignment from the ontologies (A').

To pinpoint the appropriate job for an applicant or a suitable candidate for a job opening, we needed semantic matching approaches which can deal with the highly formal human resource ontology and with the specific application requirements. The job portal developed for a particular country (e.g. Germany) must automatically compare a particular applicant profile against a multitude of job openings (or one job description against a number job seekers). For this purpose, the Matching engine uses a wired alignment (A) between the human resource ontology (O) and and the terms used in the different domains (O').

An experimental setting for this application is provided in Figure 41.1. It consists in using ontology matchers in order to generate an alignment that can be used by the matching engine in replacement of the static one and comparing the results of the whole process on some reference set.

This would provide a very precise do2002a of matcher performances in the context of the application. However, in the case of the human resource use case, this requires important modifications in the matching engine.

41.4 Summary

We have presented do2002a methods and principles for matching systems. The purpose of these tests is to allow assessing the behaviour of matching systems on a precise type of problem, i.e., when some part of the ontologies are mismatching.

This will be exploited in Section 42.3 where their results are used in order to produce profiles of matchers with regard to these characteristics.

Chapter 42

Matcher profiles

We analyse here the matchers with three different techniques in order to assess the adequation of matchers on the identified dimensions. These techniques are: literature analysis for finding the properties presented in papers (partially based on [Euzenat *et al.*, 2004a]); exploitation of questionnaires as well as by intensive collaborations with developer of matching approaches, results of the do2002as performed according to Chapter 41.

42.1 Analysis from literature

The panorama of matching systems is as large as diverse. This is not our purpose here to do yet another description of these systems. The interested reader is invited to consult [Euzenat *et al.*, 2004a; Euzenat and Shvaiko, 2007]. We here compare these systems and classify them with regard to the criteria identified in Chapter 39.

There are a number of constant features that are shared by the majority of systems. Also, usually each individual system innovates on a particular aspect. Let us summarise some global observations concerning the systems found in the literature and presented in [Euzenat and Shvaiko, 2007]:

– based on the number of systems considered, we can conclude that schema-based matching solutions have been so far investigated more intensively than the instance-based solutions. We believe that this is an objective trend, since we have striven to cover state of the art systems without bias towards any particular kind of solutions.
– most of the systems under consideration focus on specific application domains, such as books and music, as well as on dealing with particular ontology types, such as DTDs, relational schemas and OWL ontologies. Only a small number of systems aim at being general, i.e., suit various application domains, and generic, i.e., handle multiple types of ontologies. Some examples of the latter include Cupid [Madhavan *et al.*, 2001], COMA and COMA++ [Do and Rahm, 2002], Similarity Flooding [Melnik *et al.*, 2002], and S-Match [Giunchiglia and Shvaiko, 2003b].
– most of the approaches take as input a pair of ontologies, while only a small number of systems take as input multiple ontologies. Some examples of the latter include DCM [Chang *et al.*, 2005] and Wise-Integrator [He *et al.*, 2004].
– most of the approaches handle only tree-like structures, while only a small number of systems handle graphs. Some examples of the latter include Cupid [Madhavan *et al.*, 2001],

COMA and COMA++ [Do and Rahm, 2002], and OLA [Euzenat and Valtchev, 2004].

– most of the systems focus on discovery of one-to-one alignments, while only a small number of systems have tried to address the problem of discovering more complex correspondences, such as one-to-many and many-to-many, e.g., iMAP [Dhamankar *et al.*, 2004] and DCM [Chang *et al.*, 2005].
– most of the systems focus on computing confidence measures in [0 1] range, most often standing for the fact that the equivalence relation holds between ontology entities. Only a small number of systems compute logical relations between ontology entities, such as equivalence, subsumption. Some examples of the latter include CtxMatch [Bouquet *et al.*, 2006] and S-Match [Giunchiglia and Shvaiko, 2003b].

In conclusion, there is a large set of systems sharing the same type of features: schema-based matchers taking OWL or database schema as input and outputting equivalence correspondences. If this is what the application requires, then there is a wealth of systems to choose from. If the application has different requirements, there may be very few systems to choose from, if any.

Table 42.1 summarises how the matching systems cover the solution space in terms of the classifications of [Shvaiko and Euzenat, 2005]. For example, S-Match [Giunchiglia and Shvaiko, 2003b] exploits string-based element-level matchers, external matchers based on WordNet, propositional satisfiability techniques, etc. OLA [Euzenat and Valtchev, 2004], in turn, exploits, besides string-based element-level matchers, also a matcher based on WordNet, iterative fix-point computation, etc. Table 42.1 also testifies that ontology matching research so far was mainly focused on syntactic and external techniques. In fact, many systems rely on the same string-based techniques. Similar observation can be also made concerning the use of WordNet as an external resource of common knowledge. In turn, semantic techniques have rarely been exploited, this is only done by CtxMatch [Bouquet *et al.*, 2006], S-Match [Giunchiglia and Shvaiko, 2003b] and OntoMerge [Dou *et al.*, 2005]. Concerning instance-based system, techniques based on Naive Bayes classifier and common value patterns are the most prominent.

Table 42.1: Basic matchers used by the different systems.

	Element-level		**Structure-level**	
	Syntactic	**External**	**Syntactic**	**Semantic**
Hovy [Hovy, 1998]	string-based, language-based	-	taxonomic structure	-
TranScm [Milo and Zohar, 1998]	string-based	built-in thesaurus	taxonomic structure, matching of neighbourhood	-
DIKE [Palopoli *et al.*, 2003]	string-based, domain compatibility	WordNet	matching of neighbourhood	-
SKAT [Mitra *et al.*, 1999]	string-based	auxiliary thesaurus, corpus-based	taxonomic structure, matching of neighbourhood	-
Artemis [Castano *et al.*, 2000]	domain compatibility, language-based	common thesaurus (CT)	matching of neighbours via CT, clustering	-
H-Match [Castano *et al.*, 2006b]	domain compatibility, language-based, domains and ranges	common thesaurus (CT)	matching of neighbours via CT, relations	-
Anchor-Prompt [Noy and Musen, 2001]	string-based, domains and ranges	-	bounded paths matching: (arbitrary links), taxonomic structure	-
OntoBuilder [Modica *et al.*, 2001]	string-based, language-based	thesaurus look up	-	-

Table 42.1: Basic matchers used by the different systems (continued).

	Element-level		Structure-level	
	Syntactic	**External**	**Syntactic**	**Semantic**
Cupid [Madhavan *et al.*, 2001]	string-based, language-based, datatypes, key properties	auxiliary thesauri	tree matching weighted by leaves	-
COMA & COMA++ [Do and Rahm, 2002]	string-based, language-based, datatypes	auxiliary thesauri, alignment reuse, repository of structures	DAG (tree) matching with a bias towards various structures (e.g., leaves)	-
SF [Melnik *et al.*, 2002]	string-based, datatypes, key properties	-	iterative fix-point computation	-
XClust [Lee *et al.*, 2002]	cardinality constraints	WordNet	paths, children, leaves, clustering	-
CtxMatch/CtxMatch2 [Bouquet *et al.*, 2006]	string-based, language-based	WordNet	-	based on description logics
S-Match [Giunchiglia and Shvaiko, 2003b]	string-based, language-based	WordNet	-	propositional SAT
ASCO [Bach *et al.*, 2004]	string-based, language-based	WordNet	iterative similarity propagation	-
BayesOWL [Pan *et al.*, 2005]	text classifier	Google	Bayesian inference	-
Chang et al. [Chang *et al.*, 2005]	-	-	correlation mining	-
T-tree [Euzenat, 1994]	-	-	correlation mining	-
LSD/GLUE/ iMAP [Doan *et al.*, 2001; Doan *et al.*, 2004] [Dhamankar *et al.*, 2004]	WHIRL, Naive Bayes	-	-	-
Kang & Naughton [Kang and Naughton, 2003]	information entropy	-	mutual information, dependency graph matching	-
sPLMap [Nottelmann and Straccia, 2005]	Naive Bayes, kNN classifier, string-based	-	-	-
SEMINT [Li and Clifton, 1994; Li and Clifton, 2000]	neural network, datatypes, value patterns	-	-	-
Clio [Miller *et al.*, 2000; Haas *et al.*, 2005]	string-based, language-based, Naive Bayes	-	-	-
NOM & QOM [Ehrig and Sure, 2004] [Ehrig and Staab, 2004]	string-based, domains and ranges	application-specific vocabulary	matching of neighbours, taxonomic structure	-
OLA [Euzenat and Valtchev, 2004]	string-based, language-based, datatypes	WordNet	iterative fix-point computation, matching of neighbours, taxonomic structure	-
Wise-Integrator [He *et al.*, 2004; He *et al.*, 2005]	string-based, language-based, datatypes, value patterns	WordNet	clustering	-

Table 42.2 summarises the position of these systems with regard to some of the requirements of Chapter 39 (namely those requirements that can be given in the specification of the system rather than being measured). In Table 42.2, the *Input* column stands for the input taken by the systems. In particular, it mentions the languages that the systems are able to handle (if this information was

not available form the articles describing the corresponding systems we used general terms, such as database schema and ontology instead). This is, of course, very important for someone who has a certain type of ontology to match and is looking for a system. The *Needs* column stands for the resources that must be available for the system to work. This covers the automatic aspect of Chapter 39, which is here denoted by *user* when user feedback is required, *semi* when the system can take advantage of user feedback but can operate without it and *auto* when the system works without user intervention (of course, the user can influence the system by providing the initial input or evaluating the results afterwards, but this is not taken into account). Similarly, the *instances* value specifies that the system requires instances to work. Also some systems require *training* before the actual matching as well as *alignment* to be improved.

Table 42.2: Position of the presented systems with regard to some criteria of Chapter 39.

System	Input	Needs	Output	Operation
DELTA [Clifton *et al.*, 1997]	relational schema, EER	user	alignment	-
Hovy [Hovy, 1998]	ontology	semi	alignment	-
TranScm [Milo and Zohar, 1998]	SGML, OO	semi	translator	data translation
DIKE [Palopoli *et al.*, 2003]	ER	semi	merge	query mediation
SKAT [Mitra *et al.*, 1999]	RDF	semi	bridge rules	data translation
Artemis [Castano *et al.*, 2000]	relational schema, OO, ER	auto	views	query mediation
H-Match [Castano *et al.*, 2006b]	OWL	auto	alignment	P2P query mediation
Tess [Lerner, 2000]	database schema	auto	rules	version matching
MapOnto [An *et al.*, 2005a; An *et al.*, 2005b]	relational schema, XML schema, OWL	alignment	rules	data translation
Anchor-Prompt [Noy and Musen, 2001]	OWL, RDF	user	axioms (OWL/RDF)	ontology merging
OntoBuilder [Modica *et al.*, 2001]	web forms, XML schema	user	mediator	query mediation
Cupid [Madhavan *et al.*, 2001]	XML schema, relational schema	auto	alignment	-
COMA & COMA++ [Do and Rahm, 2002]	relational schema, XML schema, OWL	user	alignment	data translation
SF [Melnik *et al.*, 2002]	XML schema, relational schema	user	alignment	-
XClust [Lee *et al.*, 2002]	DTD	auto	multi-alignment	-
ToMAS [Velegrakis *et al.*, 2004]	relational schema, XML schema	query, alignment	query, alignment	data transformation
OntoMerge [Dou *et al.*, 2005]	OWL	alignment	ontology	ontology merging
CtxMatch/CtxMatch2 [Bouquet *et al.*, 2006]	classification, OWL	user	alignment	-
S-Match [Giunchiglia and Shvaiko, 2003b]	classification, XML schema, OWL	auto	alignment	-
HCONE-merge [Kotis *et al.*, 2006]	OWL	semi, user	ontology	ontology merging
MoA [Kim *et al.*, 2005]	OWL	auto	axioms OWL	-
ASCO [Bach *et al.*, 2004]	RDFS, OWL	auto	alignment	-

Table 42.2: Position of these systems with regard to the requirements of Chapter 39 (continued).

System	Input	Needs	Output	Operation
BayesOWL [Pan *et al.*, 2005]	classification, OWL	auto	alignment	
OMEN [Mitra *et al.*, 2005]	OWL	auto	alignment	-
DCM [Chang *et al.*, 2005]	web form	auto	multi-alignment	data integration
T-tree [Euzenat, 1994]	ontology	auto, instances	alignment	-
CAIMAN [Lacher and Groh, 2001]	classification	semi, instances, training	alignment	-
FCA-merge [Stumme and Mädche, 2001]	ontology	user, instances	ontology	ontology merging
LSD/GLUE [Doan *et al.*, 2001; Doan *et al.*, 2004]	relational schema, XML schema, taxonomy	auto, instances, training	alignment	-
iMAP [Dhamankar *et al.*, 2004]	relational schema, XML schema	auto, instances, training	mediator	-
Automatch [Berlin and Motro, 2002]	relational schema	auto, instances, training	alignment	-
SBI&NB [Ichise *et al.*, 2004]	classification	auto, instances, training	alignment	-
Kang & Naughton [Kang and Naughton, 2003]	relational schema	instances	alignment	-
Dumas [Bilke and Naumann, 2005]	relational schema	instances	alignment	-
Wang & al. [Wang *et al.*, 2004]	web form	instances	alignment	data integration
sPLMap [Nottelmann and Straccia, 2005]	database schema	auto, instances, training	rules	data translation
SEMINT [Li and Clifton, 2000]	relational schema	auto, instances (opt.), training	alignment	-
Clio [Miller *et al.*, 2000; Haas *et al.*, 2005]	relational schema, XML schema	semi, instances (opt.),	query transformation	data translation
IF-Map [Kalfoglou and Schorlemmer, 2003a]	KIF, RDF	auto, instances, common reference	alignment	-
NOM & QOM [Ehrig and Sure, 2004; Ehrig and Staab, 2004]	RDF, OWL	auto, instances (opt.)	alignment	-
oMap [Straccia and Troncy, 2005b]	OWL	auto, instances (opt.), training	alignment	query answering
Xu & al. [Xu and Embley, 2003; Embley *et al.*, 2004]	XML schema, taxonomy	auto, instances (opt.), training	alignment	-
Wise-Integrator [He *et al.*, 2004; He *et al.*, 2005]	web form	auto	mediator	data integration
OLA [Euzenat and Valtchev, 2004]	RDF, OWL	auto, instances (opt.)	alignment	-
Falcon-AO [Hu *et al.*, 2005]	OWL	auto instances (opt.)	alignment	-
Corpus-based matching [Madhavan *et al.*, 2005]	relational schema, web form	text corpora, instances, training	alignment	-
APFEL [Ehrig *et al.*, 2005]	RDF, OWL	user	alignment	-
eTuner [Sayyadian *et al.*, 2005]	relational schema, taxonomy	auto	alignment	-

The *Output* delivered by a system is very important because it shows the capability of the system to be used for some applications, e.g., a system delivering views and data translators cannot be used for merging ontologies. It is remarkable that many systems deliver alignments. As such, they are not fully committed to any kind of operation to be performed and can be used in a variety of applications. This could be viewed as a sign of possible interoperability between systems. However, due to the lack of a common alignment format, each system uses its own way to deliver alignments (as lists of URIs, tables, etc.). Finally, the *Operation* column describes the ways in which a system can process alignments.

Not all the requirements are addressed in Table 42.2. Indeed, completeness, correctness, runtime should not be judged from the claims of system developers. No meaningful system can be proved to be complete, correct or as fast as possible in a task like ontology matching. Therefore, the degree of fulfillment of these requirements must be evaluated and compared across systems. Moreover, different applications have different priorities regarding these requirements, hence, they may need different systems. Thus, this do2002a depends on an application in which the system is to be used.

42.2 Analysis from questionnaire

In order to collect data regarding existing matchers we have developed an online questionnaire based on the characteristics of matching approaches defined in Chapter 40. The survey allows us, on the one hand, to analyze the existing matchers, and, on the other hand, provide data important for the process of selection of the suitable matching approaches described in §43.3. Furthermore, the analysis of the data gathered enables exploitation of the valuable ideas embedded in current matching approaches and at the same time accounts for their limitations.

The matcher developers were asked to rate their algorithms by answering the 37 questions, which are divided into 8 groups:

– *introductory questions*, which are related to the developer team and its institution;

– *matcher input size-related questions* aim to collect information about the size of the input, which are served by the particular matching approach (e.g. How well does your approach support matching of sources with small number of axioms?)

– *matcher input type-related questions* gather information about the inputs the matcher handles (e.g. How well does your approach match formal ontologies?)

– *matcher approach-related questions* collect data concerning the matching approach itself (e.g. How well does your approach support black box paradigm?);

– *matcher usability-related questions* provide information regarding the applications and usage of the given matching approach (e.g. How well does your approach support sources integration?);

– *matcher output-related questions* gather data concerning the output delivered by the matching approach (How well does your approach support the global / local cardinality?)

– *matcher documentation-related questions* gather data regarding the quality of the available matcher documentation (e.g. How high is the quality of the matcher documentation?)

– *matcher costs-related questions* collect information about costs of the usage of your matching approach (e.g. How high are the costs for the matcher licence?).

The online questionnaire[1] to be filled out by the developers of the particular matchers or by the experts in the matching domain allows the addition and ranking (by using a predefined scale) of matcher alternatives. The developers weighted their algorithms according to the defined questions by using a scale between 0 and 8 (cf. Table 42.3).

Rating	**Meaning**
0	no answer / the approach does not support the feature
1	between no support and slightly support
2	slightly support
3	between slightly and well support
4	well support
5	between well and very well support
6	very well support
7	between very well and extremely well support
8	extremely well support

Table 42.3: Rating scale.

In the following sections we briefly analyze some data provided by the developers whose matcher took part in the Ontology Alignment Evaluation Initiative (OAEI) 2006 Campaign[2]. For certain characteristics we provide a diagram which shows a weighting of a particular matcher concerning the property given and within the description of the results we use the scale mentioned above.

42.2.1 Size-related questions

In the first set of questions the developers rated their approaches regarding the size of the input their matcher can handle. According to our analysis of the provided answers, all algorithms can *well* to *extremely well* deal with two incoming sources but only MOAM and HMatch are able to serve more then two ontologies *well* or *very well*, respectively.

Taking into account the number of ontological primitives within the sources all the approaches can handle *very well* to *extremely well* small (up to 100 primitives) and five of them (Falcon-AO, PRIOR, RiMOM, AOAS and HMatch) can still deal with medium (up to 500) size ontologies. The situation looks a bit different if we consider the large (up to 1,000) and extremely large (over 1,000) sources. Even these ontologies can be handled by the five approaches mentioned only PRIOR can extremely well tackle the problem while the others are only at an average (cf. Figure 42.1).

While analyzing the number of different types of ontological primitives, we found it striking that matching of ontologies, which contain axioms, is still a challenging issue since only four (Falcon, OWL-CtxMatch, RiMOM, HMatch) algorithms were able to handle this type of primitives (cf. Figure 42.2).

[1]`http://matching.ag-nbi.de`

[2]AOAS (NIH), Automs, Falcon-AO, ISLab HMatch, MAOM-QA, PRIOR, RIMOM, OWL CtxMatch

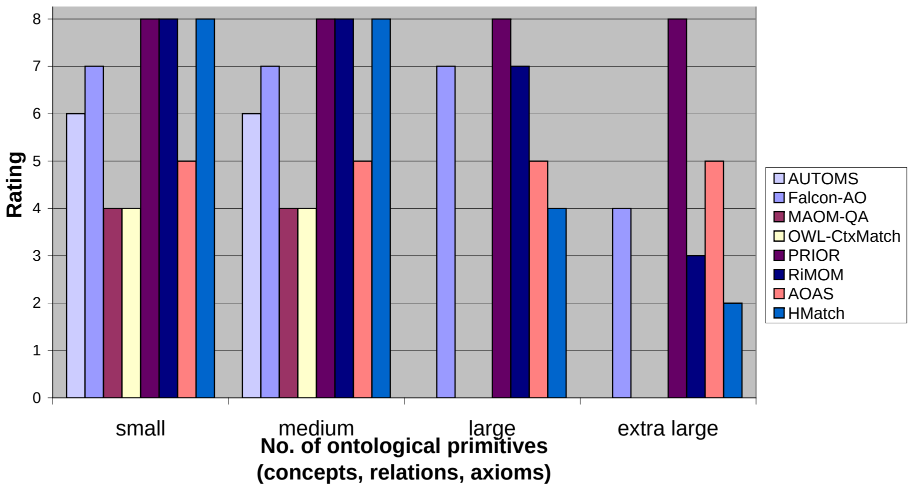


Figure 42.1: How well does the approach serve small/medium/large/extra large ontologies?

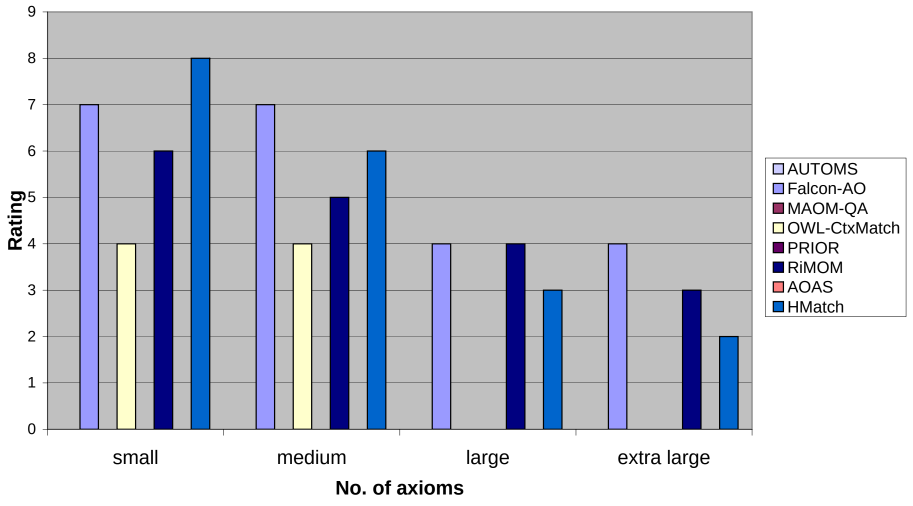


Figure 42.2: How well does the approach match ontologies with different number of axioms?

42.2.2 Input type-related questions

One of the important properties of matching approaches is the type of input on which they can operate successfully. To be able to review the "matchability" of the existing approaches regarding the heterogeneity of the incoming sources, in the subsequent set of questions we concentrated on attributes which characterize the matcher input like input category, formality level (highly/semi-informal ontology, semi-formal ontology, rigorously formal ontology, cf. §40.1), input type (instance, schema) as well as natural and representation languages.

First of all, let's analyze the existing matchers considering sources with and without instances: while almost all approaches are able to match schemes only Automs, Prior and RiMOM can handle instances. Furthermore, if we take a look at the different input categories, we notice that all algorithms can (at least well) deal with taxonomies and ontologies but only four of them (Falcon, RiMOM, AOAS, HMatch) are able to match thesauruses and, in addition, Falcon and RiMOM handle glossaries (cf. Figure 42.3).

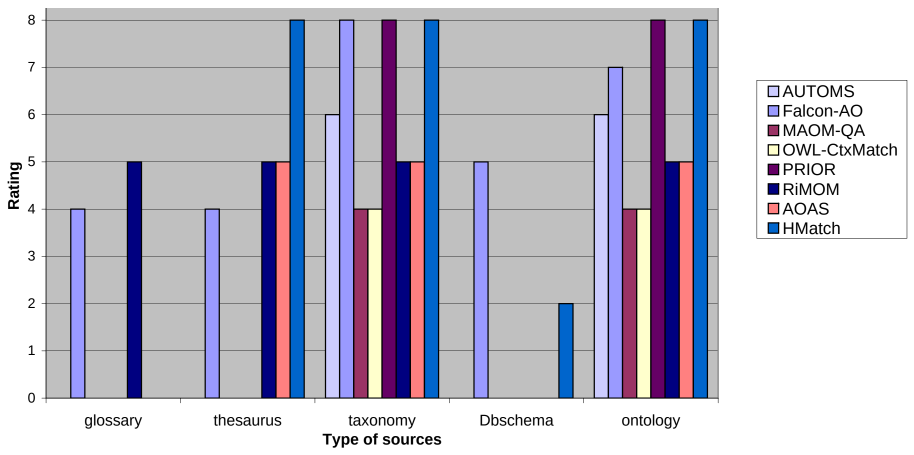


Figure 42.3: How well does the approach handle different types of sources?

Beyond this, if we analyze the heterogeneity of the input sources regarding the natural and representation languages, the best results concerning the variety of suitable matching approaches can be achieved by sources which use only one natural or representation language (Figure 42.4). Only RiMOM is a (natural) language independent approach which means that it can also serve multilingual ontologies.

42.2.3 Approach-related questions

During the ability and suitability examination of different approaches, not only the features of the input but especially the characteristics of the particular matcher itself play a crucial role. Figure 42.5 illustrates how well the approaches take into account the different processing types. While almost all systems follow the black box paradigm (i.e. automatic execution without human intervention) AOAS and RiMOM require some manual pre-processing, while the latter also needs post processing effort.

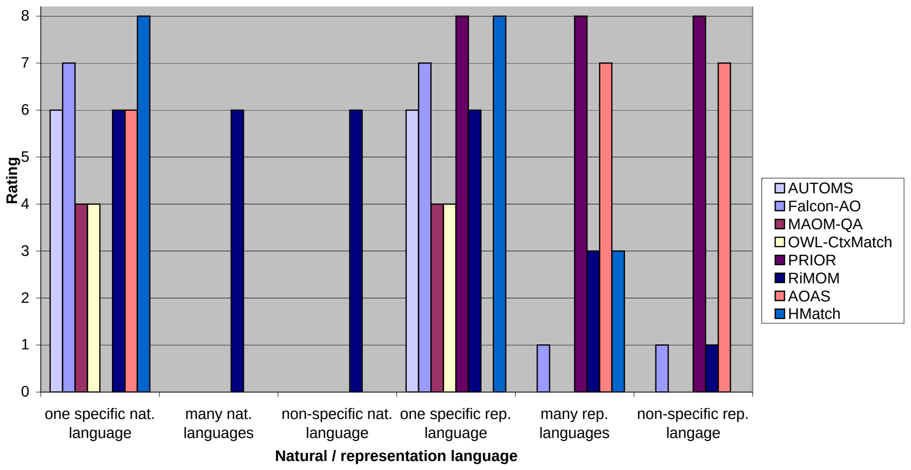


Figure 42.4: How well does the approach handle sources taking into account the rep. and nat. languages?

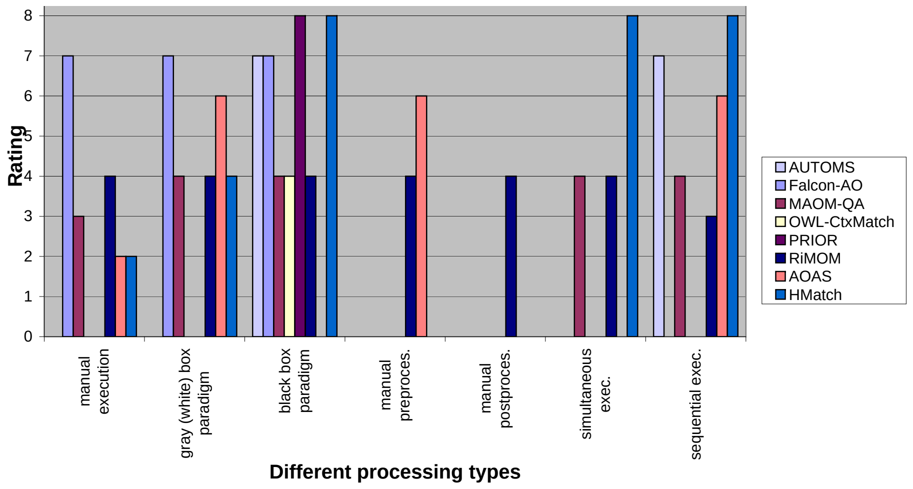


Figure 42.5: How good is the approach for different processing types?

42.2.4 Usability-related questions

Beside the features previously mentioned the matcher developers were ask to rate the approaches regarding their adaptation in the context of different application purposes (integration, translation, reuse of sources, etc.), various application goals (local use, internet use) as well as adaptation in different tasks, applications and domains. In Figure 42.6 we show as an example, how the approaches behave regarding the local, network or internet usage. Unfortunately the most of the approaches have been developed to be run on local machines and only four systems (Automs, Falcon, MAOM, and HMatch) can be accessible on a network or as an online service.

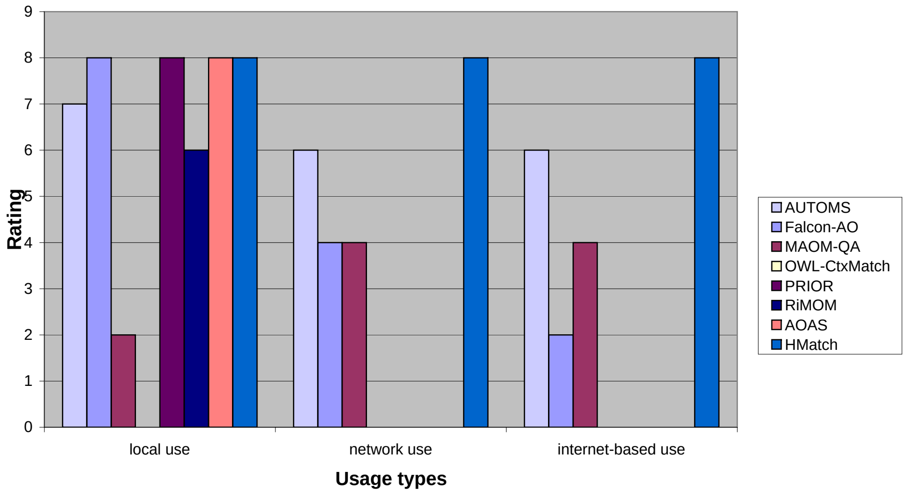


Figure 42.6: How good is the approach for different types of usage?

42.2.5 Output-related questions

The characteristics of the output is a constraint as well as those of the input. In Figure 42.7 we take a look at the output cardinality, which is one of the properties describing the output of the approaches. Falcon and HMatch (very well or extremely well respectively) support with the same intensity each type of the cardinality. Furthermore, AOAS serves only the global cardinality, RiMOM concentrates on (global and local) 1:1 while OWL-CtxMatch supports (global and local) n:m mapping.

42.2.6 Documentation-related questions

The most subjective characteristic to rate, in our opinion, is the quality and clarity of the matcher documentation. All the developers (except the OWL-CtxMatch developers) assign the rates between low (3) and high (5) for each asked aspects of the matcher documentation.

42.2.7 Conclusion

A fully developed Semantic Web will contain numerous distributed and ubiquitously available ontologies which are used in different tasks and disciplines. In turn, this means that a fundamental requirement for the realization of this vision are proved and tested ontology matching algorithms

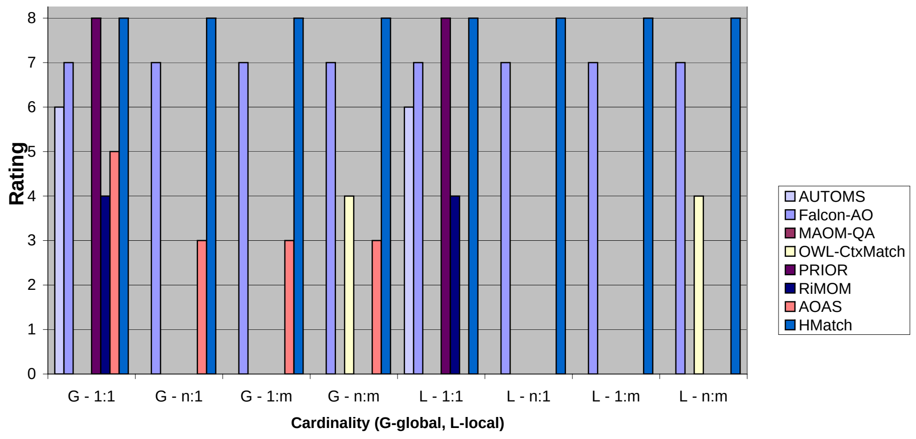


Figure 42.7: How well does the approach support global/local cardinality??

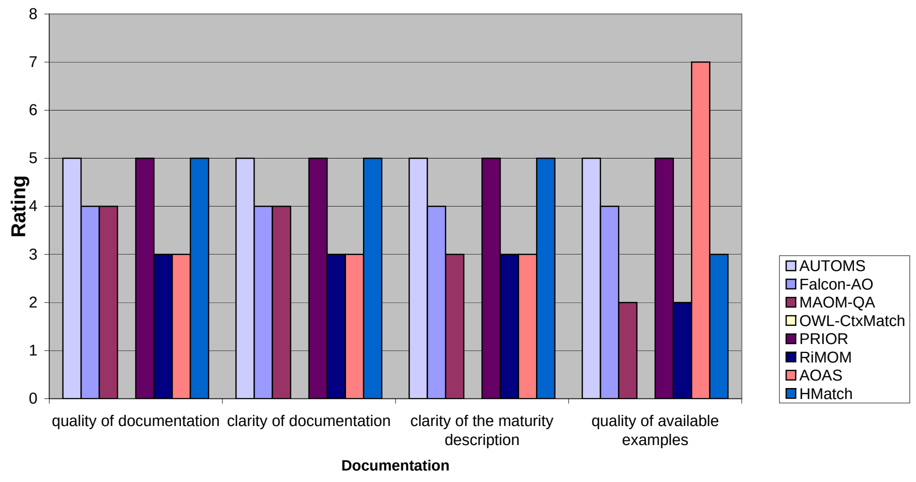


Figure 42.8: How high is the quality of the documentation?

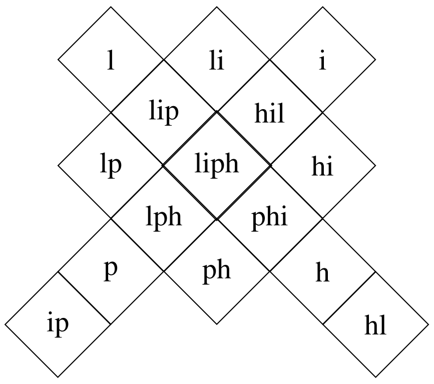


Figure 42.9: Diagram for presenting the results of matcher do2002as. Each cell corresponds to the availability of some features in the test (l=label and comments, p=properties, i=instances, h=hierarchy).

(matcher), capable of handling the heterogeneity of ontological sources available on the Web. In other words there is a need for matchers which can operate, for instance, within different contexts (i.e. different applications, tasks and domain), on different types of input (schemes, instances), on sources with various numbers of ontological primitives, and sources covering different formality levels. Taking into account the results of the survey developed, the matching of large and very large ontologies with and without axioms need to be more deeply addressed in the future. Furthermore, most matchers require specific representation and natural language, which means that approaches supporting multilingual sources and sources implemented in different representation languages are still a challenging and important issues especially with regard to the needs of people having ontologies in certain languages who wish to utilize the existing matchers for their applications.

Aside from this, Semantic Web-compatible matching methods are expected to satisfy two core requirements. First, they should be usable by ontology engineers to develop application ontologies as a combination of existing knowledge resources. This requirement is followed e.g. by the need to have, for one, a matcher output which is understandable not only to machines but also to humans, and, two, good documentation including well described examples. The second core requirement considers that matching methods are acknowledged as a core enabling technology for mediating Web Service interactions which in turn means that the approaches need to be processed automatically. Furthermore, they must be accessible not only locally but especially though the internet.

Generally speaking, despite the impressive number of research initiatives in the matching field containing valuable ideas and techniques there are still many open issues which need to be addressed in the near future.

42.3 Analysis from do2002a

We present below each system that has been evaluated with the tests of §41.2. These systems are thus able to take OWL as input and to generate alignments under the Alignment API. The presentation of each system comes either from the OM workshop proceedings (written by the participants to the do2002a) [Shvaiko *et al.*, 2006] or from [Euzenat and Shvaiko, 2007].

The results are synthesised in a diagram as explained in Figure 42.9. Each cell corresponds to a group of tests in which the ontologies to compare share a common set of characteristics (namely that the same set of features has been altered). Each cell is presented with a color representing the average of the F-measure in each of these tests. The darker the cell, the better the algorithm. The value is also presented within the diagram. Figure 42.10 shows these results for a simple measure of edit distance on class names. This shows, as expected, that such a method is only good when labels are preserved.

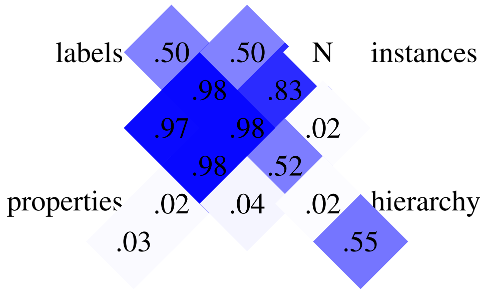


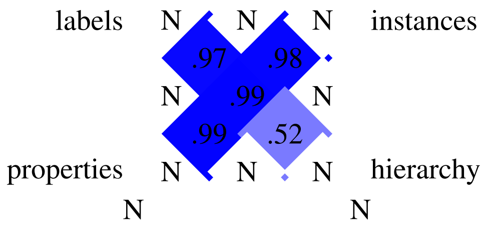
Figure 42.10: edna do2002a on F-measure (the darkest the best).

The cell in the diagram have been positioned so that the value for a cell should be related to that of its neighbours.

The use of such diagrams for an application developer consists of characterizing the cell in the diagram which correspond to the application data (by the presence/absence of labels, properties, instances or hierarchy) and to select the best matcher with regard to this cell. We applied here this technique to F-measure, however, it can be applied to precision, recall or other measures as well. Then the user can take advantage of the preference of one measure over another given in Table 43.1.

The two first systems have only a limited characterisation (restricted to 5 cells) because, they were evaluated in 2004 in which the set of tests was smaller.

42.3.1 Prompt (2004)



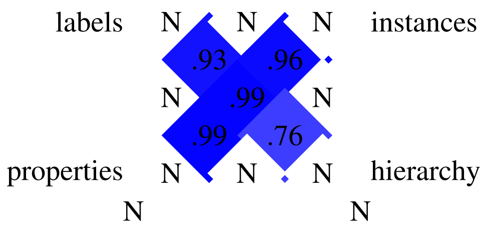
The Prompt Suite[3] is an interactive framework for comparing, matching, merging, maintaining versions, and translating between different knowledge representation formalisms [Noy and Musen, 2003; Noy, 2004]. The main tools of the suite include: an interactive ontology merging tool, called iPrompt [Noy and Musen, 2000] (formerly known as Prompt), an ontology matching tool, called Anchor-Prompt and [Noy and Musen, 2001]), an ontology-versioning tool, called PromptDiff [Noy and Musen, 2002b], and a tool for factoring out semantically complete sub-ontologies, called PromptFactor.

The Prompt have used PromptDiff when participating to the Ontology Alignment Contest of 2004. PromptDiff compares ontology versions and identifies the changes. PromptDiff produces a structural diff between two versions based on heuristics. It borrows many of them from iPrompt

[3]http://protege.stanford.edu/plugins/prompt/prompt.html

in order to identify what has changed from one version of an ontology to another. These heuristics include various techniques such as analysis and comparison of concept names, slots attached to concepts. The PromptDiff approach has two parts: (i) an extensible set of heuristic matchers; and (ii) a fixed-point algorithm which combines the results of the matchers until they produce no more changes in the diff.

42.3.2 Fujitsu (2004)

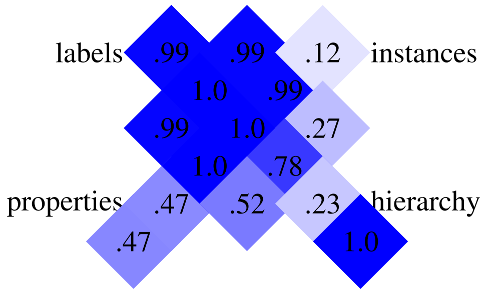


The system presented by Fujitsu implements semantic category matching (SCM) technology to the ontology alignment problem. Semantic category matching is based on a statistical approach that takes sample documents from each category and hierarchy description data, and outputs all category pairs that semantically correspond with the two classification systems. Ontology alignment is a problem designed to find couples of corresponding classes. While the purposes of SCM and ontology alignment are different, the problem structures of both are similar to each other, from the perspective of alignment between the hierarchy.

The system works through:

1. Hierarchical version of keyword extraction.
2. Similarity search category similarities, based on oblique coordinates, and
3. Structural consistency analysis.

42.3.3 OLA (2005)



OLA (OWL Lite Aligner) [Euzenat and Valtchev, 2004] is an ontology matching system which is designed with the idea of balancing the contribution of each of the components that compose an ontology, e.g., classes, constraints, data instances. OLA handles ontologies in OWL. It first compiles the input ontologies into graph structures, unveiling all relationships between entities. These graph structures produce the constraints for expressing a similarity between the elements of the ontologies. The similarity between nodes of the graphs follows two principles: (i) it depends on the category of node considered, e.g., class, property, and (ii) it takes into account all the features of this category, e.g., superclasses, properties.

The distance between nodes in the graph are expressed as a system of equations based on string-based, language-based and structure-based similarities. These distances are almost linearly aggregated (they are linearly aggregated modulo local matches of entities). For computing these distances, the algorithm starts with base distance measures computed from labels and concrete datatypes. Then, it iterates a fix-point algorithm until no improvement is produced. From that solution, an alignment is generated which satisfies some additional criterion on the alignment obtained and the distance between matched entities.

OLA also participated in 2004, but we retained the last figures, i.e., those of 2005.

42.3.4 Foam (2005)

FOAM [Ehrig, 2006], is a general tool for processing similarity-based ontology matching. It follows a general process made of the six following steps:

Feature engineering selects the features of the ontologies that will be used for comparing the entities.

Search step selection selects the pairs of elements from both ontologies that will be compared.

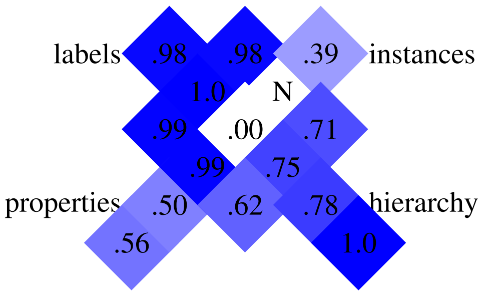
Similarity computation actually computes the similarity between the selected pairs using the selected features.

Similarity aggregation combines the similarities obtained as the result of the previous step for each pair of entities.

Interpretation extracts an alignment from the computed similarity.

Iteration iterates this process, if necessary taking advantage of the current computation.

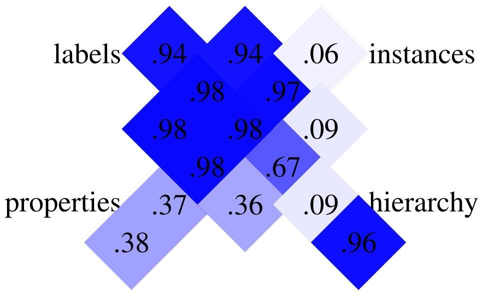
Foam adopted some of the idea of parallel composition of matchers from COMA (§42.3.10). Some innovations with respect to COMA are in the set of elementary matchers based on rules, exploiting explicitly codified knowledge in ontologies, such as information about super- and sub-concepts, super- and sub-properties, etc. At present the system supports 17 rules related to ontology features. For example, a rule states that if super-concepts are the same, the actual concepts are similar to each other. These rules use many terminological and structural techniques.



Foam also participated in 2004, but we retained the last figures, i.e., those of 2005.

42.3.5 oMap (2005)

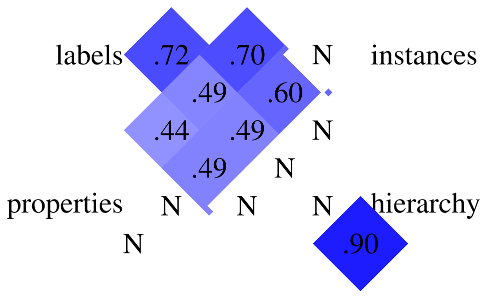
oMap [Straccia and Troncy, 2005b] is a system for matching OWL ontologies. It is built on top of the Alignment API and has been used for distributed information retrieval in [Straccia and Troncy, 2006]. oMap uses several matchers (called classifiers) that are used for giving a plausibility of a correspondence as a function of an input alignment between two ontologies. The matchers include (i) a classifier based on classic string similarity measure over normalised entity names; (ii) a Naive Bayes classifier used on instance data, and (iii) a "semantic" matcher which propagates initial weights through the ontology constructors used in the definitions of ontology entities. This last one starts with an input



alignment associating plausibility to correspondences between primitive entities and computes the plausibility of a new alignment by propagating these measures through the definitions of the considered entities. The propagation rules depend on the ontology constructions, e.g., when passing through a conjunction, the plausibiliy will be minimised. Each matcher has its own threshold and they are ordered among themselves.

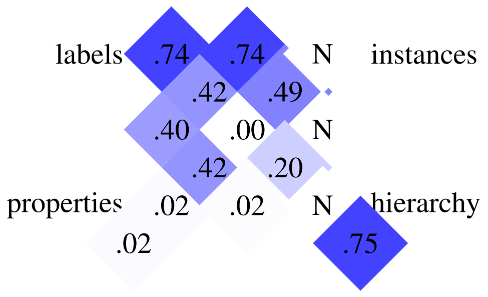
There are two ways in which matchers can work: (i) in parallel, in which case their results are aggregated through a weighted average, such that the weights correspond to the credit accorded to each of the classifiers, (ii) in sequence, in which case each matcher only adds new correspondences to the input ontologies. A typical order starts with string similarity, before Naive Bayes, and then the "semantic" matcher is used.

42.3.6 ctxMatch2 (2004)



CtxMatch2 [Bouquet *et al.*, 2003c; Bouquet *et al.*, 2003b; Bouquet *et al.*, 2006] uses the semantic matching approach. It translates the ontology matching problem into the logical validity problem and computes logical relations, such as equivalence, subsumption between concepts and properties. CtxMatch2 is a sequential system. At the element level it uses only WordNet to find initial matches for classes as well as for properties. At the structure level, it exploits description logic reasoners, such as Pellet[4] or FaCT[5] to compute the final alignment.

42.3.7 CMS (2004)



The CROSI Mapping System (hereafter, CMS) has been developed in the context of the CROSI project (which stands for Capturing Representing and Operationalising Semantic Iinteroperability). It has been evaluated at an early stage in the OAEI 2005 campaign. CMS is a structure matching system that capitalizes on the rich semantics of the OWL constructs found in source ontologies and on its modular architecture that allows the system to consult external linguistic resources.

The modular architecture of CMS employs a multi-strategy system comprising of four modules, namely, Feature Generation, Feature Selection and Processing, Aggregator and Evaluator. In this system, different features of the input data are generated and selected to fire off different sorts of feature matchers. The resultant similarity values are compiled by multiple similarity aggregators running in parallel or consecutive order. The overall similarity is then evaluated to initiate iterations that backtrack to different stages.

CMS uses various families of algorithms in order to compute similarity based on string distance. Sometimes, natural language processing methods are employed to cut down the number of string tokens that need to compute the similarity for. In particular, CMS uses tools that exploit

[4]http://www.mindswap.org/2003/pellet/

[5]http://www.cs.man.ac.uk/simhorrocks/FaCT

synonymy at the: (1) lexical-level, e.g. the use of WordNet to provide a source of synonyms, hypernyms and hyponyms; the (2) phrase- and sentence-level, e.g. phrases and sentences in the active and passive forms but having the same meanings; and the (3) semantic-level synonymy. Structural information is exploited through a sophisticated structure traversing algorithm that collects evidence of similarities along its path. Finally, CMS uses the ontology structure in different ways. Heuristic rules is the most common way to take structures into account, e.g. identifying similarity of two entities based on the status of their parents and siblings.

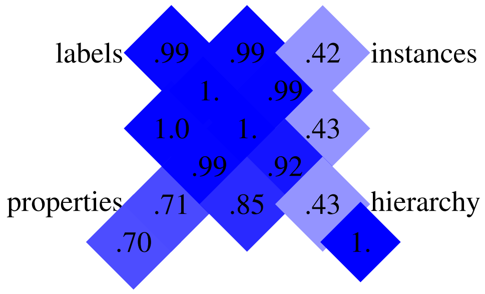
42.3.8 Falcon-AO (2006)

Falcon-AO is a system for matching OWL ontologies. It is made of two components:

LMO is a linguistic matcher. It associates with each ontology entity a bag of words which is built from the entity label, the entity annotations as well as the labels of connected entities. The similarity between entities is based on a TF/IDF computation [Qu *et al.*, 2006].

GMO is a bipartite graph matcher [Hu *et al.*, 2005]. It starts by considering the RDF representation of the ontologies as a bipartite graph which is represented by its adjacency matrix (A and A'). The distance between the ontologies is represented by a distance matrix (X) and the distance (or update) equations between two entities are simply a linear combination of all entities they are adjacent to (i.e., $X^{t+1} = AX^tA'^T + A^TX^tA'$). This process can be bootstraped with an initial distance matrix and the real process is more complex than described here because it distinguishes between external and internal entities as well as between classes, relations and instances.

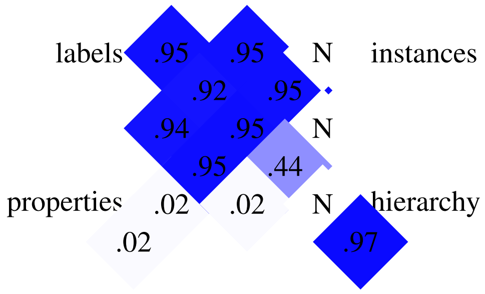
These two components are used in sequence: First LMO is used for assessing the similarity between ontology entities on the basis of their name and text annotations. If the result has a high confidence, then it is directly returned for extracting an alignment. Otherwise, the result is used as input for the GMO matcher which tries to find an alignment on the basis of the relationships between entities [Jian *et al.*, 2005].



Falcon also participated in 2005, but we retained the last figures, i.e., those of 2006.

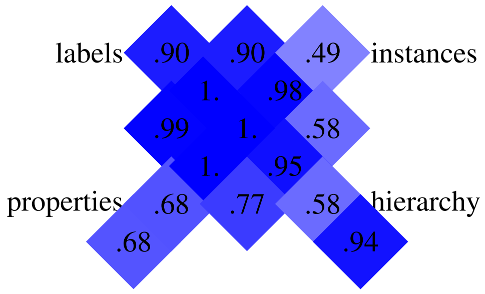
42.3.9 H-Match (2006)

H-Match [Castano *et al.*, 2006b] is an automated ontology matching system. It was designed to enable knowledge discovery and sharing in the settings of open networked systems, in particular within the Helios peer-to-peer framework [Castano *et al.*, 2005]. The system handles ontologies specified in OWL. Internally, these are encoded as graphs using the H-model representation [Castano *et al.*, 2005]. H-Match inputs two ontologies and outputs (one-to-one or one-to-many) correspondences between concepts of these ontologies with the same or closest intended meaning. The approach is based on a similarity analysis through affinity metrics, e.g., term to term affinity, datatype compatibility, and thresholds. H-Match computes two types of affinities (in the [0 1] range), namely *linguistic* and *contextual* affinity. These are then combined by using weighting schemas, thus yielding a final measure, called *semantic* affinity. Linguistic affinity builds on top of a thesaurus-based approach of the Artemis system. In particular, it extends the Artemis approach (i) by building a common thesaurus involving such relations among WordNet synsets as meronymy or coordinate terms, and (ii) by providing an automatic handler of compound terms (i.e., those composed by more than one token) that are not available from WordNet. Contextual affinity requires consideration of the neighbour concepts, e.g., linked via taxonomical or mereological relations, of the actual concept.



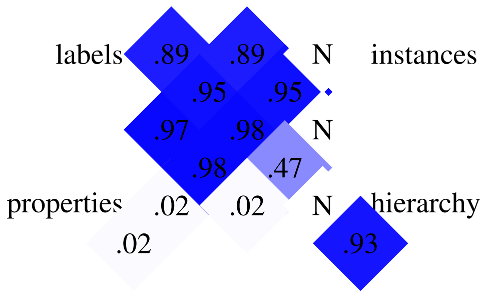
One of the major characteristics of H-Match is that it can be dynamically configured for adaptation to a particular matching task. Notice that in dynamic settings complexity of a matching task is not known in advance. This is achieved by means of four matching models. These are: *surface*, *shallow*, *deep*, and *intensive*, each of which involves different types of constructs of the ontology. Computation of a linguistic affinity is a common part of all the matching models. In case of the surface model, linguistic affinity is also the final affinity, since this model considers only names of ontology concepts. All the other three models take into account various contextual features and therefore contribute to the contextual affinity. For example, the shallow model takes into account concept properties, whereas the deep and the intensive models extend previous models by including relations and property values, respectively. Each concept involved in a matching task can be processed according to its own model, independently from the models applied to the other concepts within the same task. Finally, by applying thresholds, correspondences with semantic (final) affinity higher than the cut-off threshold value are returned in the final alignment.

42.3.10 Coma (2006)



COMA (COmbination of MAtching algorithms) [Do and Rahm, 2002] is a schema matching tool based on parallel composition of matchers. It provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for the do2002a of the effectiveness of the different matchers. As in [Do and Rahm, 2002], COMA contains 6 elementary matchers, 5 hybrid matchers, and one reuse-oriented matcher. Most of them implement string-based techniques, such as affix, n-gram, edit distance; others share techniques with Cupid (thesauri look-up, etc.). Reuse-oriented is an original matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as directed acyclic graphs, where elements are the paths. This aims at capturing contexts in which the elements occur. Distinct features of the COMA tool in respect to Cupid are a more flexible architecture and the possibility of performing iterations in the matching process. It presumes interaction with users who approve obtained matches and mismatches to gradually refine and improve the accuracy of match. COMA++ is built on top of COMA by elaborating in more detail the alignment reuse operation, provides a more efficient implementation of the COMA algorithms and a graphical user interface.

42.3.11 OWL CtxMatch (OCM, 2006)

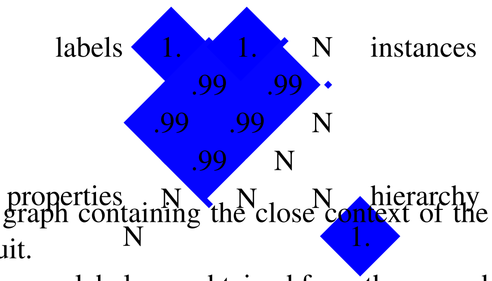


The OWL CtxMatch algorithm has been designed to match OWL DL ontologies. It is an OWL specialized version of the CtxMatch algorithm (like ctxMatch2), which is designed as a general algorithm to discover semantic relationships across distinct and autonomous generic structures. The main requirement it imposes on the structures being matched is the necessity of labelling them with natural language labels. The unique feature that both algorithms offer is that they perform so called "semantic matching" and as a result are able to recognize a broad range of relationships between matched entities, i.e. not only equivalence but also subsumption, disjointness and intersection. It assumes that the entities of given ontologies are named with commonly used words, not with some unintelligible symbols or randomly generated texts. OWL CtxMatch has already been used in one of the latest OWL-S matchmakers called Cobra1Matchmaker.

OWL CtxMatch similarly to CtxMatch realizes matching as a series of computations of relationships in which each computation is performed for every single pair of unfamiliar entities coming from both given structures. Each mapping is computed in two steps. At first internal representations of both entities are built, by means of which the algorithm stores recognized interpretations. These interpretations are defined in the form of appropriate logical formulas. In OWL CtxMatch there have been proposed description logics formulas, which are more expressive than propositional logics formulas used in original CtxMatch. The second step finds single mapping by computing what relationship holds between particular entities on the basis of their internal representations. Since OWL CtxMatch uses description logics formulas, this step in practice is realized by an external DL reasoner that initially merges both sets of formulas into one model, performs classification of it and finally determines what kind of relationship holds between counterparts for the particular entities.

The current version of the algorithm uses solely WordNet dictionary as a source of both lexical and domain knowledge and therefore is limited to the English language only. It uses Pellet as an OWL reasonner.

42.3.12 DSSim (2006)



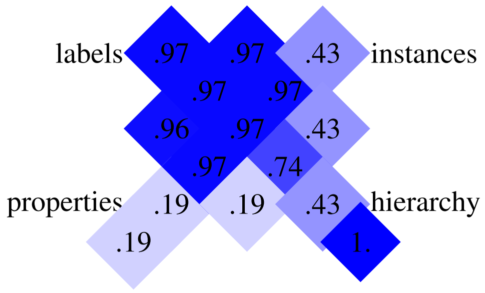
DSSim aims at producing alignments dynamically at runtime. Though, its goal is to be monitored through human interaction, the DSSim algorithm is an iterative closed loop which creates the mapping without any human interaction and works as follows:

1. For each ontology entity of the first ontology, a graph containing the close context of the entity, such as the concept and its properties, is buit.
2. Syntactically similar entities or those bearing synonym labels are obtained form the second ontology and a graph containing them is built.
3. The similarity between nodes is assessed through different similarity algorithms and their output is combined using the Dempster rule of combination.

4. The similiarity pairs bearing the highest belief function are selected and added to the alignment.

In order to avoid a complex graph of relationships, a limit on the number of synonyms, which are extracted from the WordNet, is defined.

42.3.13 Automs (2006)



Automs is the successor of the HCONE-merge system. HCONE-merge is an approach to domain ontology matching and merging exploiting different levels of interaction with a user [Kotis *et al.*, 2006; Vouros and Kotis, 2005; Kotis and Vouros, 2004]. First, an alignment between the two input ontologies is computed with the help of WordNet. Then, the alignment is processed straightforwardly by using some merging rules, e.g., renaming, into the new merged ontology. The HCONE basic matching algorithm works in six steps:

1. Chose a concept from one ontology.
2. Retrieve the WordNet senses of this concept;
3. Retrieve hypernyms and hyponyms of these senses.
4. Exaluate for the most frequent terms within the vicinity (senses, hponyms and herperonyms) their frequency of occurence.
5. Build a query from the related concepts related to the initial concept in the input ontology.
6. Use Latent Semantic Indexing for determining the best sense to be used in the query.

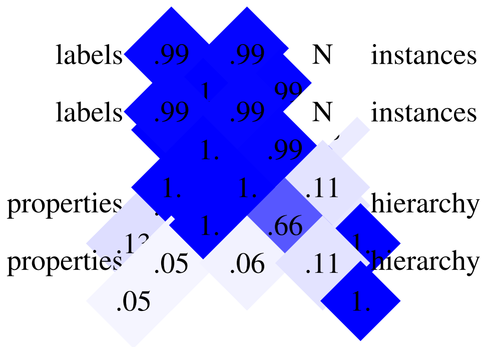
The highest graded sense expresses the most possible meaning for the concept under consideration. Finally, the relationship between concepts is computed. For instance, equivalence between two concepts holds if the same WordNet sense has been chosen for those concepts based on six steps procedure described above. The subsumption relation is computed between two concepts if a hypernym relation holds between the corresponding WordNet senses of these concepts. Based on the level at which the user is involved in the matching process, HCONE provides three algorithms to ontology matching. These are: fully automated, semi-automated and user-based. The user is involved in order to provide feedback on what is to be the correct WordNet sense on a one by one basis (user-based), or only in some limited number of cases, by exploiting some heuristics (semi-automated).

42.3.14 Onto-Mapology (2006)

Onto-Mapology is the Johns Hopkins University Applied Physics Lab (JHU/APL) ontology mapping software solution that was designed and developed with strong consideration for human participation in the mapping process. It integrates techniques based on string and text matching, graph structure matching, and rule-based semantic matching. It allows users to apply different combinations of these techniques, or a hybrid algorithm. This system is dedicated to OWL ontologies.

String matching is based on off-the-shefves solutions and in particular, Jaro-Winkler similarity, 2-gram based simiilarities and an information retrival indexing tool. Structural matching then compare the ontology entities in function of their neighbours matching or not (starting form the string matching results). If two entities have a large proportion of neighbours matchings, then they are considered matching. Finally, Onto-Mapology expresses matching rules that are applied to the current match in order to improve it (generally by providing more matches). This is called semantic matching.

42.3.15 Prior (2006)



The Prior system generalises the notion of virtual document used in Falcon-AO In Prior, the profile of a concept is a combination of all linguistic information of the concept, i.e. the profile of a concept = the concept's name + label + comment + property restriction + other descriptive information. The Profile Enrichment is a process of using a profile to represent a concept in the ontology, and thus enrich its information. The purpose of profile enrichment is based on the observation that though a name is always used to represent a concept, sometimes the information carried in a name is restricted. While, other descriptive information such as comments may contain words that better convey the meaning of the concept.

Profile Propagation exploits the neighboring information of each concept. The profile information is spread to ancestors, children or siblings of the ontology entities. The propagation is associated with weights which follows two rules: (1) The closer the concepts, the higher the weights, and (2) The weight of a parent is higher than the weight of a child and the weight of a child is higher than the weight of a sibling.

From these profiles, Prior uses cosine similarity in the vector space associated to profiles (as bag of words). Simultaneously, the String Mapper computes the similarity between the names of different concepts using Levenshtein distance. Both similarity are combined. Alternatively, when ontologies are too large, Prior uses an information retrieval system (Indri) each entity of each ontology are considered as queries against the other ontology and the top-ten answers are preserved. A Mapping Extractor extracts matched entities from the candidate matched entities and outputs the results.

42.3.16 RiMoM (2006)

RiMOM is a tool for ontology alignment by combining different strategies. Each strategy is defined based on one kind of information or one type of approach. In its current version, there are five strategies: edit-distance based strategy, statistical-learning based strategy, and similarity-propagation strategies.

There are six major steps in the alignment process of RiMOM:

1. Similarity factors estimation. Given two ontologies, it estimates two similarity factors, which respectively approximately represent the structure similarity and the label similarity of the two ontologies. The two factors are used in the next step of strategy selection.

2. Strategy selection. The basic idea of strategy selection is if two ontologies have high label similarity factor, then RiMOM will rely more on linguistic based strategies; while if the two ontologies have high structure similarity factor, then we will employ similarity-propagation based strategies on them.
3. Strategy execution. The selected strategies are used independently to find the alignments. Each strategy outputs an alignment.
4. Alignment combination. The alignment are combined through a linear-interpolation method.
5. Similarity propagation. If the two ontologies have high structure similarity factor, RiMOM employs an algorithm called similarity propagation to refine the found alignments and to find new alignments that can not be found using the other strategies. Similarity propagation makes use of structure information.
6. Alignment refinement. The resulting alignments are improved through several heuristic rules to remove the "unbelievable" alignments.

RiMOM offers various possible strategies such as linguistic based strategies (edit-distance and K-Nearest Neighbor statistical-learning strategies) and structural propagation strategies (concept-to-concept propagation strategy (CCP), property-to-property propagation strategy (PPP), and concept-to-property propagation strategy (CPP)).

Depending on their label and structure similarity factors, the algorithm will favour one or the other king of strategy. For this purposes it uses heuristic rules. For example, if the structure similarity factor is lower than some threshold (.25) then RiMOM does not use the CCP and PPP strategies. However, the CPP will always be used in the alignment process.

These characterisation of the evaluated algorithms can be particularly useful because they are very precise on what kaind of input the algorithms can handle well (or relatively well). It is clear that most algorithms are able to do very well when the labels remain unaltered, then when labels are modified – which is most of the real matching tasks – algorithms perform more or less correctly. We will see later how to take advantage of this.

42.4 Summary

In summary, these do2002as are very complementary: an application developer should first select the systems that can handle its data (input and output types). This can be achieved from the literature survey or the result of the questionnaire. Then it should find the best performer depending of her needs and the characteristics of her data. This can be achieved by looking closely at the cell that corresponds to the kind of data that has to be used by the system.

Most of the criteria are not comparable for various reasons. So it is difficult to compare the information gathered with the three techniques above. Moreover, information coming from these three sources can be useful for choosing a system. It is thus necessary to combine them when matching them with the application requirements.

This is considered in the next chapter.

Chapter 43

Finding suitable matchers

When the matcher profiles have been obtained and the application characteristics are known, it is necessary to match them in order to decide which matcher to use. This can be done through various methods. The difficulty comes from the large amount of non comparable characteristics to take into account.

In this chapter we provide two methods for supporting the decision. The first one (Section 43.1) is a very simple method that is based on the characterization of applications in large classes like in Chapter 39. The second one is a more elaborated framework that uses all the characteristics provided in Chapter 40 and offers a method for multi-criteria decision making called Analytic Hierarchy Process (AHP)(cf. Section 43.2) applied to the matching selection process (cf. Section 43.3).

43.1 Balancing criteria by aggregating measures

[Ehrig, 2006] provided an analysis of the different needs for do2002a depending of the considered applications. We have applied his technique to the requirement table (Table 39.1) of Chapter 39 [Euzenat and Shvaiko, 2007]. As a matter of fact, it can be rewritten in function of the measurements obtainable by evaluating the matchers. We used this technique to design Table 43.1. Therefore, different *application profiles* could be established to explicitly compare matching algorithms with respect to certain tasks.

Such a table can be useful for aggregating the measures corresponding to each of these aspects with different weights or to have an ordered way to interpret do2002a results.

Different measures suit different do2002a goals. If we want to improve systems, it is best to have as many indicators as possible. However, in order to single out the best system, it is generally better to focus on the very relevant factors. This can be achieved by only selecting the few very relevant factors, e.g., speed and precision for semantic web browsing, or by aggregating measures in relation with their relevance.

For aggregating measures depending on a particular application, its is possible to use weights corresponding to the values of Table 43.1, and thus respecting the importance of each factor. Weighted aggregation measures (weighted sum, product or average) can be used.

F-measure is already an aggregation of precision and recall. It can be generalised as a harmonic mean, for any number of measures. This requires to assign every measurement a weight, such that these weights sum to 1. Obviously the weights have to be chosen carefully, again depending on

Application	speed	automatic	precision	recall
Ontology evolution	medium	low	high	high
Schema integration	low	low	high	high
Catalog integration	low	low	high	high
Data integration	low	low	high	high
P2P information sharing	high	low	medium	medium
Web service composition	high	high	high	low
Multi agent communication	high	high	high	medium
Context matching in ambient computing	high	high	high	medium
Query answering	high	medium	medium	high
Semantic web browsing	high	medium	high	low

Table 43.1: Application requirements of Table 39.1 reinterpreted as measurement weights.

the goal.

Definition 102 (Weighted harmonic mean)**.** *Given a reference alignment R, a set of measures $(M_i)_{i\in I}$ provided with a set of weights $(w_i)_{i\in I}$ between 0 and 1 such that their sum is 1, the weighted harmonic mean of some alignment A is given by*

$$H(A,R) = \frac{\prod_{i\in I} M_i(A,R)}{\sum_{i\in I} w_i \cdot M_i(A,R)}.$$

Example 7 (Weighted aggregation of do2002a measures)**.** *Consider that we need to choose among two available systems S_1 and S_2, for an application to peer-to-peer system, semantic web browsing or schema integration. We will apply weights corresponding to the criteria of Table 43.1. The weights will be $high = 5$, $medium = 3$ and $low = 1$. They are normalised (so as to sum to 1.) for each kind of application. The performance of the systems with regard to the criteria are given in the following table as well as the resulting harmonic means for each pair system/application:*

					S_1	S_2
				Automatic	*1.*	*1.*
				Speed	*.6*	*.7*
				Recall	*.6*	*.8*
	Automatic	*Speed*	*Recall*	*Precision*	*.6*	*.5*
Peer-to-peer system	*.08*	*.42*	*.25*	*.25*	*.62*	*.67*
Semantic web browsing	*.3*	*.1*	*.1*	*.5*	*.68*	*.64*
Schema integration	*.08*	*.08*	*.42*	*.42*	*.62*	*.64*

Those who need a matching system for a peer-to-peer system or schema integration should choose system S_2 ($.67$ and $.64$ are better than $.62$) and those who want to use it for semantic web browsing should use system S_1 ($.68$ is better than $.64$).

43.2 A method to identify suitable matching approaches

As long as decisions rely on single criterion that serves as the basis for comparison of alternatives or the scales of the different criteria are consistent and numeric measures accurately capture expected performance, summary statistics or, in some cases, just acting on the human instinct may be sufficient for the decision making process. However, when the decision depends on multiple criteria and scales are not consistent the process becomes complex and difficult, and the involvement of qualitative as well as quantitative methodologies or tools is indispensable. Consequently, in such cases a multi criteria decision making process is required, otherwise known as a *Multi Criteria Decision Analysis (MCDA)*, which is a procedure that aims to support decision makers whose problems are concerned with numerous and conflicting criteria. Such methods developed for better model decision scenarios vary in their mathematical rigor, validity, and design [Grandzol, 2005]. One of such methods, a methodology for supporting a decision making process called *Analytic Hierarchy Process (AHP)* takes into account the considerations of Hahn [Hahn, 2002] regarding the need for a structured results-based approach for decision making that allows trade-offs into the systematic method, including all perspectives and considerations.

The AHP is a systematic approach developed to structure the expectance, intuition, and heuristics based decision making into a well-defined methodology on the basis of sound mathematical principles [Bhushan and Rai, 2004].

AHP provides a mathematically rigorous application and proven process for prioritization and decision-making which helps setting priorities and to making the best decision when both qualitative and quantitative aspects of a decision need to be considered [Saatly, 1990]. By reducing complex decisions to a series of pair-wise comparisons and then synthesizing the results, decision-makers arrive at the best decision based on a clear rationale. It is generally accepted, that AHP constitutes one of the best options to aid multi-criteria decision making[1]. It compares the relative importance that each criterion has with respect to the others, while enabling the relative weight of the criteria to be calculated. Finally it normalizes the weights in order to obtain the measures for the existing alternatives. The AHP-method consists of:

1. **defining the problem or the project objectives**: e.g. buying a car;

2. **building a hierarchy of decision:** AHP provides a means to break down the problem into a hierarchy of subproblems (hierarchy of goal, criteria, sub-criteria and alternatives) which can more easily be comprehended and subjectively evaluated [Bhushan and Rai, 2004]. At the root of the hierarchy is the goal (e.g. suitable car) or objectives of the problem in question, the leaf nodes are the alternatives (e.g. Mercedes, VW) which are to be compared and between these two levels are various criteria (c) and sub-criteria (sc) (e.g. c-car comfort: sc-air condition, sc-leather seat; c-car security: sc-ABS, sc-airbag and c-car body design).

3. **collecting data**; data is collected from domain experts corresponding to the hierarchy in the pairwise comparison of the alternatives on a qualitative scale. This step assesses the characteristics of each alternative (e.g. Alternative 1 (Mercedes) is much better than Alternative 2 (VW) w.r.t leather seats, airbag and car body design but Alternative 2 (VW) is better than Alternative 1 (Mercedes) considering ABS and air condition).

[1]It does not use the normalized groups of separate numbers which destroy the linear relationship among them [Fenton and Pfleeger, 1996]

4. **building a pairwise comparison:** for each level of criteria (sub-criteria and criteria) a pairwise comparison between the sibling nodes is to be built and organized into square matrix[2] (e.g. car security is much more important than car body design and more important than car comfort while car comfort is only a little bit more important than car body design).

5. **calculating the final result:** the ratings of each alternative (cf. step 3) is multiplied by the weight of the sub-criteria (cf. step 4) and aggregated to get local ratings with respect to each criterion. The local ratings are then multiplied by the weights of the criteria (cf. step 4) and aggregated to the global ratings. The final value is used to make a decision about the problem defined in the step 1.

43.3 Applying analytic hierarchy process to matcher selection

AHP is to be applied to the selection of matching approaches. By reducing complex decisions, i.e. which matching is suitable for a given set of requirements, to a series of pair-wise comparisons (dimensions, factors and attributes) and synthesizing the results (list of possible algorithms) decision-makers arrive at the best decision (the best matching approach) based on a clear rationale [Saatly, 1990].

In the following we give a brief overview of how the AHP steps described in the Section 43.2 can be applied to the process of matcher selection taking into account some tool support for the data collection and calculation of the best alternative.

1. The problem to be solved: "Which matching approach is currently relevant w.r.t the given application requirements?"

2. The hierarchy of decision is built using the taxonomy described in Section 40 whereby the goal is to "find a suitable approach" (level 0) which is connected though three levels of criteria: 1^{st}level - dimensions, 2^{nd} level - factors and 3^{rd} level - attributes with the alternative matching approaches (cf. Figure 43.1).

3. In order to collect data about the different alternatives of matching approaches and to be able to conduct the pairwise comparisons we firstly need the relevant information about the particular alternatives. For this reason we have developed (following the hierarchy of the matching characteristic) an online questionnaire (filled out by the domain and matching experts) that allows the addition and rating (by usage of a predefined scale from 0 to 8) of matching alternatives. When a matcher is added via the questionnaire into the collection of the alternatives, all available alternatives in the system are automatically weighted against the new approach. Given two matcher alternatives m_1, m_2 and criteria c as well as the user defined weighings for the single approach $w(c)_{m_1}$ and $w(c)_{m_2}$ the weighings for the pairwise comparisons (between alternatives m_1, m_2) $w(c)_{m_1,m_2}$ and $w(c)_{m_2,m_1}$ are calculated as follows: (i) $w(c)_{m_1,m_2} = w(c)_{m_1} - w(c)_{m_2}$; (ii) $w(c)_{m_2,m_1} = w(c)_{m_2} - w(c)_{m_1}$.

4. Since the users of the AHP-tool have defined the requirements of their application w.r.t the suitable matching approach, they are able to weight the criteria (dimensions, factors and attributes) in the pairwise comparison on the scale from 0 - equal (two criteria have the same importance) to 8 - extremely important (one criteria is much more relevant than the

[2]For details see [Saatly, 1990]

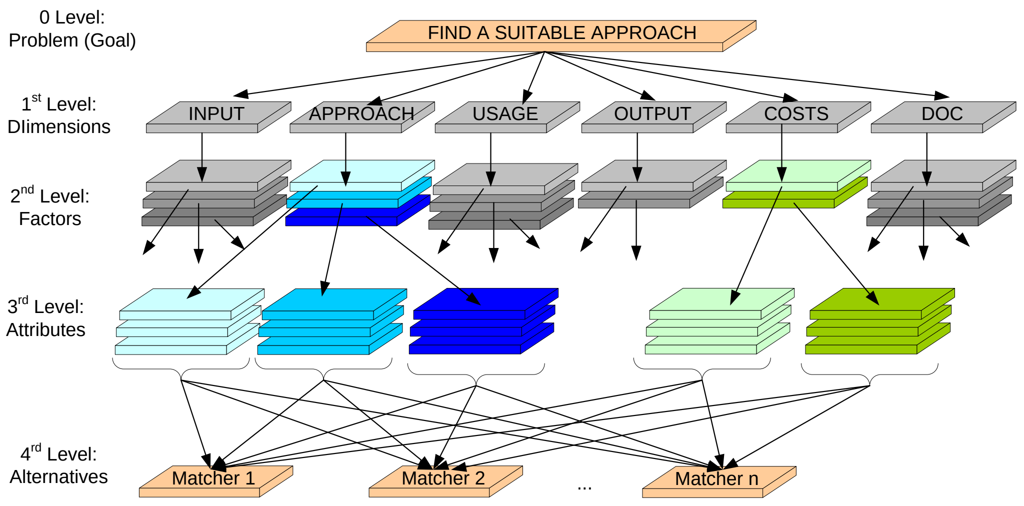


Figure 43.1: AHP hierarchy (Detection of the suitable matching approach)

other) concerning their system specification. This means, that for each level of criteria the users build a pairwise comparison between the sibling nodes: they weight the attributes against attributes, factors against factors and dimensions against dimensions (e.g. within the factor *formality level* the attribute *formal* (ontologies) is more important than *informal*, cf. Figure 43.3). If there is more then one level of criteria (as in our case), for each sub-criteria on the lowest level an "aggregated judgement" must be intended. The aggregated judgement in the hierarchy of the decision problem along all possible paths are multiplied with another from the highest to the lowest level.

Since the weighings of the criteria are the crucial part in the AHP approach they need to be deeply considered and demand the detailed analysis of the application requirements. Even if the complexity of the weighings increase with each level of criteria and it is especially challenging to rate the general dimensions, in our opinion the approach gives a relative easy to use method to find the suitable matcher by defining and rating the application requirements.

5. The decision regarding the determination of the suitable matching approach defined in the step 1 is based on the ranking $r(goal)$ of a matcher alternative m. The ranking is based on the overall priority for each alternative that can be better understood if we think of the priority for each criterion as a weight that reflects its importance. The overall priority is found by multiplying the products of the criterion priority with the priority of its decision alternative. This means that the ranking reflects the global importance of the approach according to the alternative weightings performed in step 3 as well as criteria weightings from step 4 and is calculated as follows:

$$c_{crit} = \{n | n \text{ child of } crit\}$$

$$r(crit) = \begin{cases} |\text{getWeight}(m, crit)|, & \text{if crit is at lowest hierarchy level} \\ \sum\limits_{n \in c_{crit}} r(n) \cdot |\text{getWeight}(m, n)|, & \text{otherwise} \end{cases}$$

The higher a matcher alterative m is weighted for various criteria, with each criteria weighted with respect to the users requirements, the higher the priority of the particular approach in the entire ranking. Following this weighting process the AHP tool supports the creation of a ranking of the alternatives in depending upon the multilevel hierarchy matcher characteristics, weightings of these characteristics as well as weightings of the alternatives that shows the priority of each alternative for the defined goal.

43.4 Implementation

In step 3, PHPSurveyor[3] has been used for providing the online questionnaire[4]. The collected data regarding the matcher alternatives from the questionnaire is stored in a questionnaire database (MySQL) while an additional database (AHP database) stores the weighting results of the pairwise comparisons (cf. Figure 43.2).

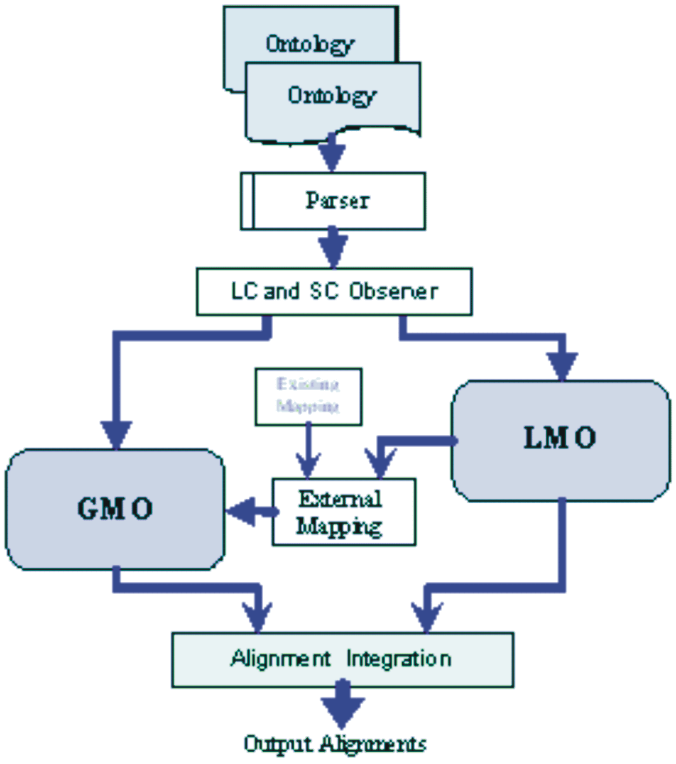


Figure 43.2: AHP Tool with online questionnaire

In step 4, to enable a user-friendly pairwise comparison of the criteria from the multilevel hierarchy matcher characteristic we developed a tool which supports the processing of the AHP method[5].

[3]http://www.phpsurveyor.org
[4]http://matching.ag-nbi.de/
[5]AHP tool is a modification of the Java AHP tool JAHP; [6]

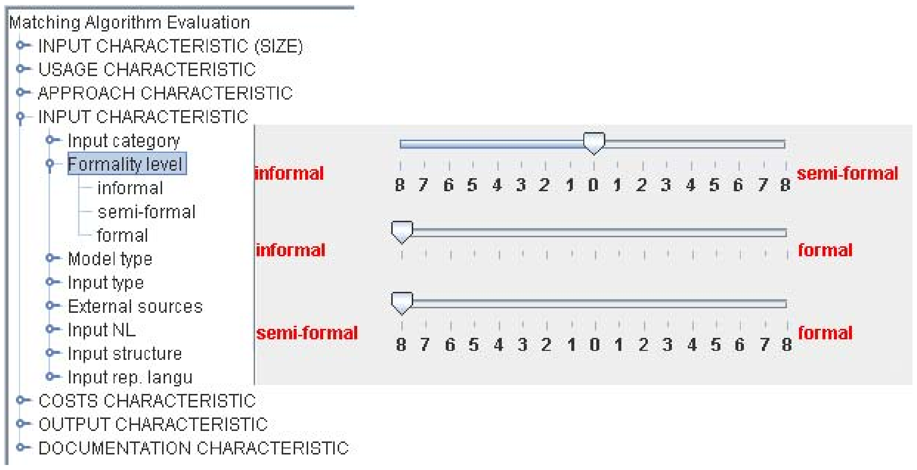


Figure 43.3: AHP tool: weighed attributes

43.5 Summary

We have presented two approaches for matching matchers to applications. The first approach is very simple but only requires few information like the one provided in Chapter 39) to provide results. The second approach if more elaborate but requires much more information, especially in ranking the characteristics.

The proposed adaption of the Analytic Hierarchy Process (AHP) to the detection of a suitable matching approach has been based on *multilevel characteristics for matching approaches* and supported by the AHP tool. We hope that by implementing tools for helping domain experts with poor expertise in ontology matching field to use the AHP, we will contribute help accurate description of the application requirements and finding appropriate approach for matching.

We will find the results in Chapter 44.

Chapter 44

Recommendations

Recommending a particular system for some application seems now possible. We apply below the techniques that have been proposed before. As mentioned, these techniques will provide results at different levels of granularity. They will thus be applied in an increasing precision order. The first results are very general since they apply to classes of applications (§44.1.1). We will thus apply them to Knowledge web use cases in order to apply more suited systems to use cases (§44.1.2). The second results use the Analytic Hierarchy Process to decide which matcher is the best for Knowledge web use case 1: Recruitments.

44.1 General view of systems and use cases

This section is a straightforward application of the techniques provided above to the results of systems evaluated during the OAEI campaigns. The results provided here should be taken as an illustration of the techniques rather than a definitive guide to matching systems. In particular, the results are only based on the system participating in OAEI, they do only take precision, recall and automatic into account and finally the weights of the measures are arbitrarily fixed.

We proceed in two steps: first we select systems for applications depending on application requirements and system measured performances. Then we apply the findings to Knowledge web use cases depending on their identified class of application.

44.1.1 Characterisation of system applicability

For each kind of application as considered in §43.1, we aim at evaluating matchers with regard to application requirements. Unfortunately, we have only few parameters that can be taken into account: automaticity requirement which is always satisfied by all the tested systems, precision and recall. We weight these three criteria as suggested in Example 7 in §43.1, i.e., $high = 5$, $medium = 3$, $low = 1$ and normalise them so that they sum to 1. Then, weighted harmonic means is computed from the performance results obtained through do2002a.

The result is provided in Table 44.1.

Unfortunately, either systems are very homogeneous and non specialised or these three criteria are not sufficient to introduce variability in choice. Indeed, the best system with regard to the measures is always the same: RiMOM.

Application	automatic	precision	recall	System
Ontology evolution	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
Schema integration	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
Catalog integration	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
Data integration	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
P2P information sharing	.14	.43	.43	RiMOM (.91), Coma (.88), Falcon (.87)
Web service composition	.45	.45	.09	RiMOM (.87), Falcon (.83), Coma (.82)
Multi agent communication	.38	.38	.23	RiMOM (.88), Falcon, Coma (.84)
Context match in ambient computing	.38	.38	.23	RiMOM (.88), Falcon, Coma (.84)
Query answering	.27	.27	.45	RiMOM (.90), Falcon, Coma (.87)
Semantic web browsing	.33	.56	.11	RiMOM (.88), Falcon (.84), Coma (.83)

Table 44.1: Advised systems given application requirements.

Use case	Type	System
Recruitments	Data integration	RiMOM (.91), Falcon, Coma (.88)
B2C marketplace for tourism	Query answering	RiMOM (.88), Falcon (.84), Coma (.83)
News aggregation	Semantic web browsing	RiMOM (.88), Falcon (.84), Coma (.83)
Real estate management	P2P information sharing	RiMOM (.91), Coma (.88), Falcon (.87)
Integrated access to biological data	Data integration	RiMOM (.91), Falcon, Coma (.88)

Table 44.2: Knowledge web use cases requiring matching systems and the advised systems according to Table 44.1

44.1.2 Application to use cases

These results can be applied directly to the Knowledge web use cases (Table 44.2). yielding first advice to system designers. Since our results are very similar, this analysis is not particularly specific. It certainly could be better if we had some measure of the speed at which these systems return results.

44.2 Application to use case 1

In this section we briefly describe how we have applied the AHP-based methodology for matcher selection described in Sec.43.3 to use case 1 situated in the human resource domain.

Since the goal of the AHP-based methodology has already been defined in §43.3 (goal: finding a suitable matching approach), the decision hierarchy regarding matcher suitability has been built (§40) and the information considering the matcher approaches has been collected during the OAEI 2006 Campaign (§42.2), the missing link in the process of matcher selection is the weightings of requirements regarding the human resources use case.

44.2.1 Pairwise comparison of the criteria

To identify the requirements of the human resources application we have analyzed the the given human resource ontology along with the restrictions regarding the suitable matching approaches[1] . In order to compare the application requirements, we first need to “translate” the relevant parts of the use case description (in the text above we have marked in bold important information) into the corresponding terms from the developed multilevel characteristic of matching approaches (cf. §40). The translation of some of the information is shown in Table 44.3

Description from human resources use case - free text	**MCMA-based description**
General information	
ontology	input characteristic: input category: ontology
domain relevant terms	input characteristic: input model type: domain ontology
domain specific knowledge	input characteristic: external sources: domain specific resources
highly formal	input characteristic: input formality level: formal ontology
OWL	input characteristic: input formality level: formal ontology
over 8.000 concepts	input characteristic: input size: number of concept: extra large input characteristic: input size: size of input: extra large
less than 100 properties	input characteristic: input size: number of relation: medium input characteristic: input size: number of relation: extra large
different properties	input characteristic: input structure: heterogeneous relations
does not contain any axioms	input characteristic: input size: number of axioms: no axioms
job postings with candidate profiles	input characteristic: input size: number of ontologies: two
data describing job applications and job offers	input characteristic: input size: number of instances: large input characteristic: input size: number of instances: extra large
automatically compare	approach characteristics: kind of similarity relation: black box paradigm
particular country	input characteristic: natural language: one natural language
job portal	usage characteristics: usage goal: internet-based use
particular applicant profile against a multitude of job openings	output characteristics: matching cardinality: 1:m

Table 44.3: HR-use case in free text and MCMA description

To define the requirements of the human resources application within context of the AHP-based methodology concerning the specification of the potential suitable matching approaches we must weight the particular properties (from the same level in the hierarchy) in the pairwise comparison on a scale from 0 to 8 (cf. Table 44.4).

Figure 44.1 shows the weighting of the attributes `local use`, `network use` and `internet-based use` within the factor `usage goal` which belongs to the dimension `usage characteristic`.

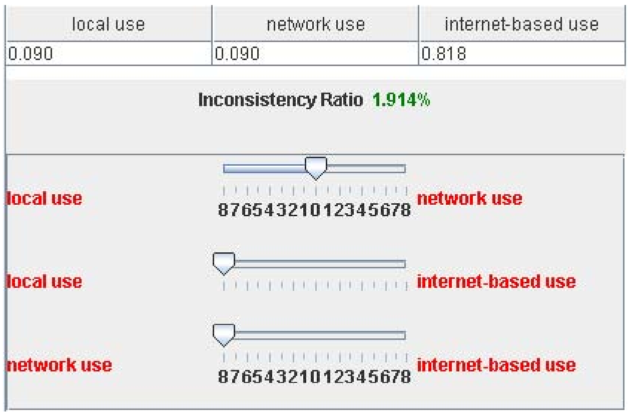
44.2.2 Results of the application of the AHP-based methodology

After weighing all the relevant (regarding the information extracted from the human resources test case description) dimensions, factors and attributes the AHP tool (cf. Sec 43.4) calculates the

[1]Unfortunately, the corresponding ontologies cannot be exemplified for confidentiality reasons

Importance	**Definition**	**Explanation**
0	equal importance	two criteria (c_1 & c_2) have the same importance
2	moderate importance	c_1 is weakly more relevant than c_2
4	strong importance	c_1 is strongly more relevant than c_2
6	very strong (demonstrated) importance	c_1 is very strongly more relevant than c_2
8	absolute importance	c_1 is absolutely more relevant than c_2
1,3,5,7	intermediate between values (for compromise between the values mentioned above)	to interpolate a compromise judgment because there is not good word to describe it

Table 44.4: AHP scale

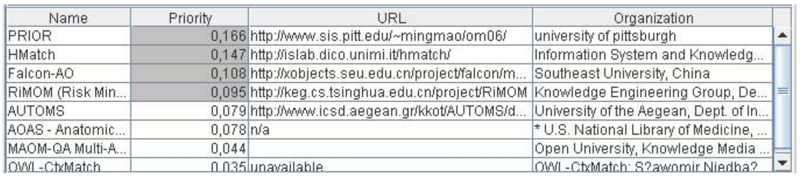


local use	network use	internet-based use
0.090	0.090	0.818

Inconsistency Ratio 1.914%

Figure 44.1: Weighting of the attributes

results and delivers the ranking of the most suitable matching approaches. The approaches which gained top ratings in the human resources test case are PRIOR, HMatch and Falcon, ranked 1^{st}, 2^{nd} and 3^{rd} respectively (cf. Figure 44.2).



Name	Priority	URL	Organization
PRIOR	0,166	http://www.sis.pitt.edu/~mingmao/om06/	university of pittsburgh
HMatch	0,147	http://islab.dico.unimi.it/hmatch/	Information System and Knowledg...
Falcon-AO	0,108	http://xobjects.seu.edu.cn/project/falcon/m...	Southeast University, China
RiMOM (Risk Min...	0,095	http://keg.cs.tsinghua.edu.cn/project/RiMOM	Knowledge Engineering Group, De...
AUTOMS	0,079	http://www.icsd.aegean.gr/kkot/AUTOMS/d...	University of the Aegean, Dept. of In...
AOAS - Anatomic...	0,078	n/a	* U.S. National Library of Medicine, ...
MAOM-QA Multi-A...	0,044		Open University, Knowledge Media ...
OWL -CtxMatch	0,035	unavailable	OWL -CtxMatch: S?awomir Niedba?

Figure 44.2: Results achieved by AHP-based mythology for the human resources use case

If we consider the terms defined in Table 44.3 and then compare the results obtained by the AHP-based methodology with the information on the diagrams from §42.2 it becomes obvious why just these approaches achieved such high rankings. For example with regard to the `input-related questions` concerning the dealing with different types of sources (cf. Figure 42.3) all three matchers are capable of handling, very well to extremely well, the ontological input type, and in addition, PRIOR is the best candidate to serve the extra large ontologies. Apart from that, concerning the `approach-related questions`, PRIOR, HMatch and Falcon have been developed to conduct automatic matching process based on the black-box paradigm (cf. Figure 42.5) while HMatch is also the best approach if internet-based usage of the approach is required.

44.3 Summary

In summary, the two do2002a procedures provide very different results. Indeed, these procedures manipulate totally different data in radically different ways. The first approach has only considered a limited set of parameters that can be relatively objectively evaluated, but used somewhat arbitrary ranking of these parameters. The second approach has used well-proved techniques for decision making on a wide range of parameters, but only questionnaire data on a limited set of systems as entry.

So, while the first approach is certainly more accurate in terms of the comparability of the data it provides and also its validity, the second approach is broader in the number of parameters it takes into account. This would certainly be improved by using the data of the first approach in the process of the second one.

Moreover, this shows that it remains difficult to evaluate the suitability of systems to application and application classes. Hence a really valuable do2002a has to go through application specific do2002a.

Chapter 45

Conclusions

Selecting an ontology matching system given a particular application is a difficult problem. Even when the application needs are very precise, there are many criteria that can be used for choosing the adequate matchers and all criteria cannot be assessed in the same way. It is also difficult to obtain all the information about each matcher (the information is most often fragmentary).

This deliverable can be seen as a first attempt at finding a way to solve these problems and providing the invaluable data for choosing a matcher. We summarise what has been done (§45.1) before providing a methodological guide for taking advantage of the findings (§45.2) and finally consider comparable work (§45.3).

45.1 Summary

In order to help identifying matching solutions to particular applications, we have performed the following steps:

– Identify relevant criteria for general categories of applications;
– Identify relevant characteristics of matchers;
– Explain how to benchmark matching systems in order to assess some of these characteristics;
– Provide profiles from a number of existing matching systems depending on the identified characteristics;
– Provide a method and present a tool for identifying suitable matching systems for an application;
– Finally, apply these techniques to Knowledge web use cases (and especially one of these).

45.2 Methodological guide

Given some application to develop, there can be several ways to use this deliverable. We characterise them as superficial, deep and involved. They are as follows:

superficial The superficial method would use the application analysis table (Table 43.1)for identifying the profile of the application and would use this either to select a subset of matchers on which to perform a deep analysis or to select the best suited matcher according to Chapter 44.

deep The deep analysis would use the tool presented in §43.2, input the detailed criterion preferences in order to find the matching system the closest to the requirements.

involved The involved approach would instrument the application in order to carry out application specific do2002a as presented in Chapter 41. This is a very costly approach however.

In the current state of the art, choosing a matching system cannot be done on a catalogue. This requires a precise analysis of the application requirement and a comparison with precise matcher characteristics. Both resources are expensive to produce.

Future work will be dedicated to the collection of further matcher alternatives (with help of the online questionnaire) and the application of the AHP tool into the various Semantic Web scenarios connected with the do2002a of the entire framework.

45.3 Related work

The closest work that we are aware of has been carried out within the INTEROP network of excellence [Huza *et al.*, 2006]. It aims at developing a system, OntoMas[1], for helping and teaching how to carry out matching. Since the tool aims at helping user to find an adequate matcher for some task, it has developed a classification of tools and a characterization of tasks. The tool classification based on [Shvaiko and Euzenat, 2005] is comparable to the one presented here and the positioning of tools within this classification is assessed through questionnaires as well. The description of the task is more rudimentary (in particular, it is based only on the syntactic aspects of the input ontologies). This is certainly because the tool must be usable by novice users who do not know the task in depth. Contrary to the work presented here, the multicriteria decision is based on ad hoc rules. Some of these rules are filters that eliminate some unsuitable systems; some others increase or decrease the score of the considered methods (usually by one point). So this scheme is again equivalent to an equi-weighted linear aggregation.

As far as we can tell, no other work has considered proved multicriteria decision techniques for matching matchers to matching tasks.

[1]http://www.polytech.univ-nantes.fr/ontomas/

Part VI

Format

Chapter 46

Introduction

In a general sense, an ontology alignment can express any kind of semantic relationship existing between the entities of two ontologies[1]. The difficulty of representing semantic relations is increased in heterogeneous knowledge-based systems like the semantic web because of the diversity of ontology representation languages and differences in conceptualisation, scope, scale, or granularity. So we motivate the need for an expressive alignment language with several ontology mediation scenarios in § 46.2, and a concrete example that focus on two wine-related ontologies, in § 46.3. This example illustrates the expressiveness requirements for alignment languages. Based on these, Chapter 50 defines the abstract syntax and model theoretic ontology semantics of such a language. Chapter 51 offers two operational syntax for the language that are used in tools implemented it and presented in Chapter 52.

46.1 Motivations

This definition of an alignment is rather abstract and does not provide a concrete format that can be used for expressing these alignments. "Reifying" alignments in a standardised format can be very useful in various contexts:

- for collecting hand-made or automatically created alignments in libraries that can be used for linking two particular ontologies;
- for modularising matching algorithms, e.g., by first using terminological alignment methods for labels, having this alignment agreed or amended by a user and using it as input for a structural alignment method;
- for comparing the results with each others or with possible "standard" results;
- for generating from the output of different algorithms, various forms of interoperability enablers. For instance, one might generate transformations from one source to another, bridge axioms for merging two ontologies, query wrappers (or mediators) which rewrite queries for reaching a particular source, inference rules for transferring knowledge from one context to another.

The goal of this deliverable is to propose some language to express these reified alignments so that they can be exchanged between applications.

[1]Ontology is here used in a broad sense: they can be databases, terminologies, as well as ontologies. Concerning the language syntax presented here, the only condition is that the entities be accessible by URI. With regard to semantics, their ontology languages must have a model-theoretic semantics.

We claim that alignments are more intelligible than transformations: they only express correspondences between ontology entities, not the way they must be used (as a transformation of ontologies, a query transformation or a data translation?). This can be the basis for studying their properties (moreover, these properties can also be inferred from the methods used for generating alignments).

The problem is thus to design an alignment format which is general enough for covering most of the needs (in terms of language and alignment output) and developed enough for offering the above functions.

46.2 Ontology mediation scenarios

Ontology mediation as a generic term gathers a set of techniques needed to achieve interoperability in semantically-enabled systems. In [Scharffe, 2005], a set of use cases of ontology mediation is identified as follows:

Query rewriting: An application uses data described according to a source ontology o and has to answer a query q written in terms of o. It may however need to evaluate the query against data described using another ontology o', e.g., by querying another source. The query q needs to be rewritten in a query q' expressed in terms of o'. In order to achieve this, a query rewriting system needs the *correspondences* existing between the concepts and properties defined in o and o'. They are obtained as an alignment A through a matching process (Matcher) as illustrated in Figure 46.1. A generator provides a query $mediator$ from A which is able to transform q into a query q' expressed with regard to o'. Once the query rewritten and addressed to the target database, the resulting instances may have to be processed using instance transformation and instance mediation techniques described in the two following paragraphs.

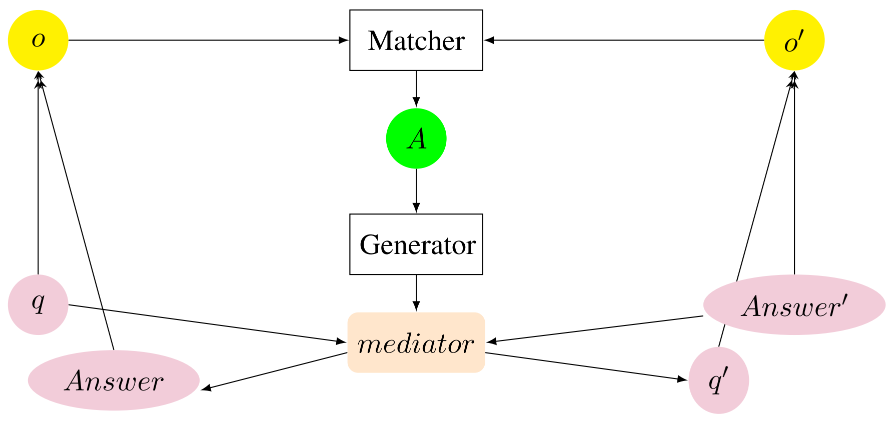


Figure 46.1: Query rewriting illustration (inspired from [Euzenat and Shvaiko, 2007]).

Instance translation: Instance descriptions, i.e., assertions about instances involving concepts and properties of one ontology o, may have to be translated for being considered in the context of an application using another ontology o'. This is typically the case in semantic web service composition. The instance translation process depicted in Figure 46.2 uses the alignment A between o and o'. This supposes that the instance translation process can be derived from this alignment. We refer the reader to § 46.3.3 for a concrete example on instance transformation. Once the instances translated, an operation must be performed to verify if there are some duplicates between the local and transformed instances, i.e., if one of the transformed instances from o is not equivalent to an already existing instance in o'. This operation is called instance mediation.

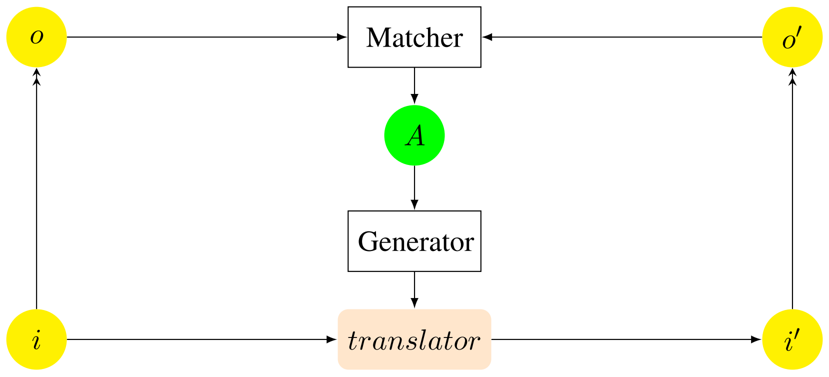


Figure 46.2: Instance translation illustration (inspired from [Euzenat and Shvaiko, 2007]).

Instance mediation: Comparing two instances in order to find out if they must be interpreted as the same individual, and eventually unifying them, is the goal of instance mediation (also called instance identification or record linkage in databases). In that perspective we can distinguish between two phases in the instance mediation process: first the instances must be recognised to be referring to the same individual; once two instances are identified as being the same, they need to be merged in one new instance combining the properties of both. Instance mediation is necessary in a query rewriting scenario once the target ontology queried and the set of returned instances transformed in terms of the source ontology. Figure 46.3 illustrates the instance mediation process. It takes advantage of the alignment A between ontology o and o' to help identifying instances. This can happen when the alignments provides correspondences between classes (or tables) and properties involved as keys.

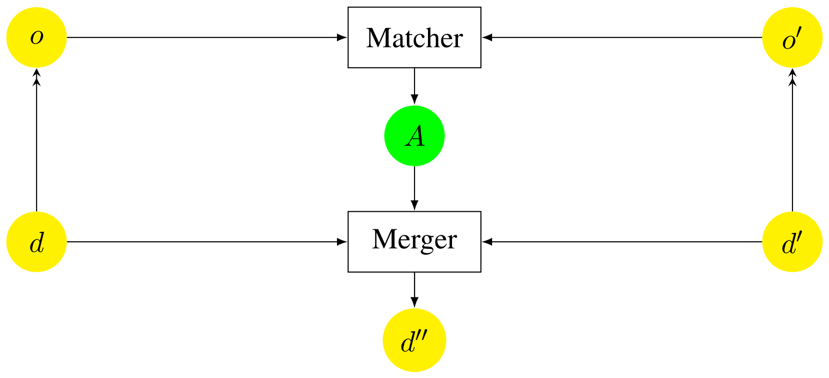


Figure 46.3: Instance mediation illustration.

This set of use cases shows the centrality of alignments in ontology mediation. Further such examples are provided in [Euzenat and Shvaiko, 2007]. The Web Ontology Language OWL [Dean and Schreiber (eds.), 2004] includes a few constructs to specify equivalence or subsumption between classes and relations but as we will see in § 46.3 these are not sufficient to specify the complex correspondences that can arise when aligning two ontologies. The next section provides examples of such correspondences.

46.3 Motivating examples

This section shows, from an example, what kind of correspondences must be expressible between ontological entities. We align the popular "Wine ontology"[2], with the "Ontologie du vin"[3]. We will refer to these two ontologies respectively by `Wine` and `Vin` in the following. `Wine` is written in OWL, while `Vin` is written in WSML, the Web Services Modeling Language [Lausen *et al.*, 2005]. The ontologies both represent important properties related to wine like the geographic origin, colour, taste, and the type of grapes used to make it, and so forth. As trying to align the two ontologies, we show that simple one-to-one correspondences between entities are not sufficient to represent the full alignment between these ontologies.

This alignment could be necessary in a scenario of a wine trading service for example, gathering data from multiple semantically described information sources.

46.3.1 Are you more Red or White wine? (Subsumption Relations)

There is an obvious equivalence correspondence between each main concept of the two ontologies both representing a wine. This correspondence would have easily been detected by an algorithm using a multilingual component by looking at the concepts labels: "`Wine`" in `Wine` and "`Vin`" in `Vin`. However, the multiple properties of each concept are requiring a deeper investigation and sometimes more complex relations than simple one to one equivalent correspondences.

The colour of a wine is in `Wine` represented through the class "`WineColor`" itself defined by three instances: "`White`", "`Rose`" and "`Red`". In `Vin` the class "`Type`" has nine instances describing the type of a wine. Red, Rose and White are among these instances together with other types of less common wines. In `Wine` the property "`hasColor`" relates a wine to its colour as defined above, while in `Vin` the property "`type`" has this role. There is a subsumption correspondence between these two properties as there is a subsumption correspondence between "`WineColor`" in `Wine` being subsumed by "`Type`" in `Vin`. For the alignment to be complete the corresponding instances need to be related. The instances left over in "`Type`" are part of a domain which is not represented in `Wine`. Figure 46.4 shows part of the set of correspondences we have just defined.

We provide the translation of these correspondences in first-order logic to help readers understanding the meaning of these pictures. Like in alignments, only the correspondences, i.e., arrows and frames, are provided in this format. The ontologies themselves, e.g., subsumption assertions, are not part of the alignment. These first-order logic expressions quatify over a set of individuals whose interpretation is assumed common, e.g., $\forall x, Wine(x) \Leftrightarrow Vin(x)$ means that any x classified as $Wine$ in the first ontology, is to be classified as Vin in the second one and vice versa.

[2]Accessible at `http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#`

[3]Accessible at `http://sw.deri.org/simfrancois/ontologies/OntologyDuVin.wsml`

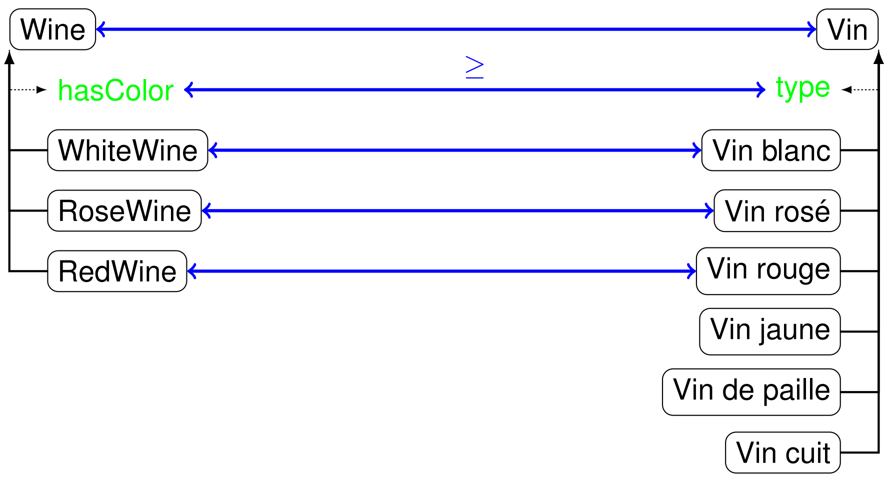


Figure 46.4: Correspondences between wines and their type or colour.

In the present case, these correspondences can be trivially transcribed as:

$$\begin{aligned}\forall x, [Wine(x) &\Leftrightarrow Vin(x)]\\ \forall x, [WhiteWine(x) &\Leftrightarrow Vin\ blanc(x)]\\ \forall x, [RoseWine(x) &\Leftrightarrow Vin\ rosé(x)]\\ \forall x, [RedWine(x) &\Leftrightarrow Vin\ rouge(x)]\\ \forall x, y, [hasColor(x, y) &\Leftarrow type(x, y)]\end{aligned}$$

46.3.2 Bordeaux wine is the best (Value Restriction)

We illustrate, in the following, a need to restrict the set of entities entering the scope of a particular correspondence. A Bordeaux wine is modelled as an instance of the class "`BordeauxWine`" in `Wine`, while in `Vin` the property "`terroir`"[4] relates a wine to an instance of the "`Terroir`" concept. To specify a correspondence between wines elaborated in the Bordeaux region we need to restrict the scope of the `Vin` class to those instances which are in the relation "`terroir`" with values `Bordeaux`.

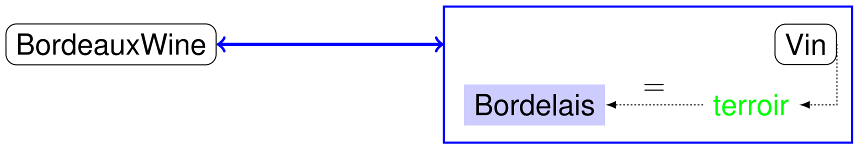


Figure 46.5: Correspondence between wines from Bordeaux.

In predicate calculus, this can be expressed by:

$$\forall x, [BordeauxWine(x) \Leftrightarrow Vin(x) \wedge terroir(x, < \#Bordelais >)]$$

[4]*Terroir* is a French word for a particular agricultural region.

with $< \#Bordelais >$ a particular individual of ontology o'.

Note that there is another kind of compound object that we call context. There are subtle differences between the two expressions of Figure 46.6:

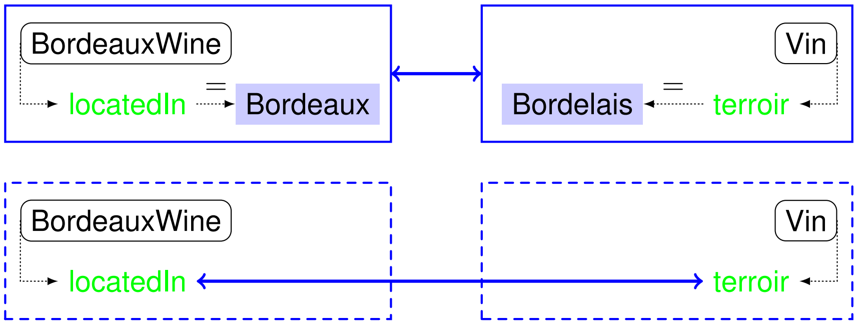


Figure 46.6: Two kinds of compound object put in correspondence: constraint on matched objects and context of matched objects.

The first part expresses that `Wines` located in `Bordeaux` are equivalent to `Vin` whose `terroir` is the `Bordelais` area. By contrast the second part expresses that in the context of `Wines` and `Vin`, `locatedIn` is equivalent to `terroir`.

The expression of the first part corresponds to the expression of Figure 46.5, while the second part can be expressed as:

$$\begin{aligned}&\forall x, [(BordeauxWine(x) \wedge Vin(x)) \\ &\quad \Rightarrow (\forall y, locatedIn(x, y) \Leftarrow terroir(x, y)) \wedge (\forall y, locatedIn(x, y) \Rightarrow terroir(x, y))]\end{aligned}$$

Of course, these two kinds of objects can be mixed in the same correspondence.

46.3.3 But quality also depends on the vintage year (Type Conversion)

An important aspect of a wine is the year it has been worked out. Both ontologies represent this data using a class named “`VintageYear`” in `Wine` and “`Millésime`” in `Vin`. Both classes have a property pointing to the value of the year. However in `Wine` the year is modelled using an integer while in `Vin` a date type is used. The correspondence should indicate in this case what is the relations between the instance values, i.e., which transformations are required to correctly map equivalent values.

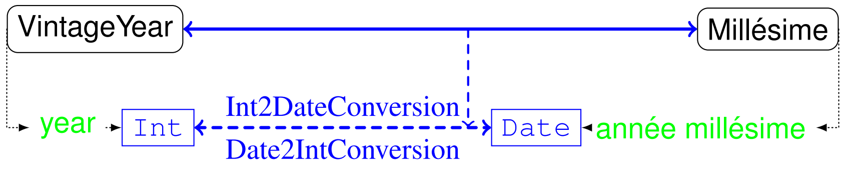


Figure 46.7: Correspondence between wine vintage years.

This can be expressed as the following:

$$\begin{aligned}&\forall x, [(VintageYear(x) \Leftrightarrow Mill\acute{e}sime(x))\\&\quad \wedge(VintageYear(x) \wedge \exists y : Int; year(x, y) \Rightarrow ann\acute{e}emill\acute{e}sime(x, Int2DateConversion(y))\\&\quad \wedge(Mill\acute{e}sime(x) \wedge \exists y : Date; ann\acute{e}emill\acute{e}sime(x, y) \Rightarrow year(x, Date2IntConversion(y))]\end{aligned}$$

46.3.4 Some may prefer locally grown wines (Path equations)

Some customers prefer wine that is bottled and sold directly from the producer (“`LocallyGrownWine`”). Our `Vin` ontology does not feature this category. However, it could be possible to match this `LocallyGrownWine` class with a restriction of the `Vin` class, namely that its `propriétaire` (owner) and `négociant` (first seller) are the same.

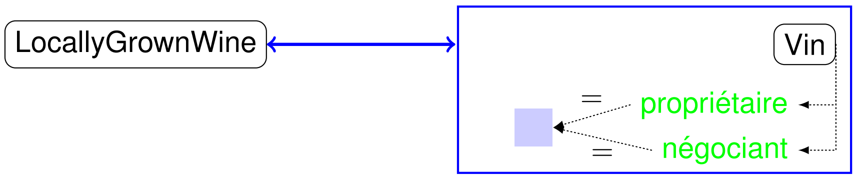


Figure 46.8: Correspondence with constraint relations.

This can easily be rendered as:

$$\forall x, [LocallyGrownWine(x) \Leftrightarrow (Vin(x) \wedge \exists y; [propri\acute{e}taire(x, y)) \wedge n\acute{e}gociant(x, y)])]$$

The correspondences presented in this section are natural and can be expected to occur frequently when mediating between ontologies. Even if they are easy to understand, they require the use of an expressive formalism in order to be expressed correctly.

46.4 Synthesis

Uncoupling alignments from ontologies brings a solution to the integration of the heterogeneous ontologies described using different formalisms. But it also requires expressiveness: all examples above are calling for an expressive alignment language. In deliverable 2.2.6, we surveyed existing such languages. We do not reproduce this survey here. It suggested, in particular, to merge the Ontology Mapping Language [Scharffe and de Bruijn, 2005] developed by the Ontology Management Working Group, and the Alignment format [Euzenat, 2004] developed by INRIA. This would benefit from the expressiveness of the OMWG Mapping language and the openness of the Alignement API.

Both languages are already independent from concrete ontology representation languages but provided different facilities: the Alignment format has been designed as an extensible framework for expressing alignments offering an operational implementation for manipulating the alignments while the OMWG format is an expressive alignment format requiring more complex parsing and rendering procedures. The Alignment format is used in several systems and in particular in the OAEI initiative while the OMWG format is used in the WSMT tool for web service manipulation.

Merging both formats allows expressive OMWG alignments to be considered as Alignments in the Alignment format and benefits from all the tools built around the Alignment API. On the opposite, the Alignment API can take advantage of an expressive format.

This deliverable presents a language, result of the integration of both formalisms, for expressing all kinds of alignments presented in this chapter independently from any ontology representation language. This language is presented through its syntax and semantics. It is illustrated through the examples given before and the implementation of this language is finally described.

46.5 Requirements

The requirements that can be put on the format to be defined are the following:

– being Web ready: in particular using URIs and semantic web languages (XML, RDF);
– being language independent: this allows alignments between ontologies written in different languages;
– being simple so that current ontology matching tools can manipulate it without having to implement a full-fledged knowledge representation language;
– being expressive so that it can cover an important part of the usable relations between ontologies;
– supporting many different uses: in particular not being committed to one particular usage (being used as axioms in one language, or being used for some particular transformation);
– supporting as many different kinds of manipulation (trimming, composing, etc.) as possible.

[Euzenat, 2004] proposed an alignment format and an application programming interface (API) for manipulating alignments, illustrated through a first implementation. This first implementation offered a relatively limited expression language but provided many support function. It has been used in the Ontology Alignment Evaluation Initiative campaigns as well as in other tools.

A number of other formats have been proposed for expressing alignments in the context of numerous applications. The purpose of this deliverable is to define an alignment format that can satisfy most of the needs of existing formats.

46.6 Document outline

In the next chapter, the various formats that can be used for expressing alignments in a declarative manner are presented. Chapter 48, then evaluates the similarity and differences between these formats in order to better understand the need of such a format. We demonstrate how some of these formats are able to achieve limited interoperability with others (mostly through export/import functions). In the last chapter we specify a common format which is based on both the Alignment format and SEKT Mapping language and which can be the basis for an expressive and independent alignment format.

Chapter 47

Existing formats

At the beginning of Knowledge web there were very few formats (in fact only the SBO was documented). Independently, many different efforts proposed their own format or at least what could be considered an alignment format. We briefly present here the various formats that have been proposed so far for expressing relations between ontologies on the world wide web. We mostly compare these formats based on their syntax. Each section ends with a conclusion that summarises the benefits and drawbacks of the formats from that standpoint.

A deeper analysis of some of these in terms if semantics and expressiveness are provided in deliverable 2.2.5 [Hitzler *et al.*, 2005a].

47.1 OWL

OWL in itself can be considered as a language for expressing correspondences between ontologies. As a matter of fact, the `equivalentClass` and `equivalentProperty` primitives have been introduced for relating elements in ontologies describing the same domain. This use is even documented by the W3C best practices working group [Uschold, 2005]. However, these primitives are only shorthands for other primitives (i.e., `subClassOf`, `subPropertyOf`) that already allow to relate entities. So the following OWL ontology:

```
<owl:Class rdf:about="http://ian.ac.uk#Nappy">
  <owl:equivalentClass rdf:resource="http://jim.us#Diaper"/>
</owl:Class>

<owl:Class rdf:about="http://ian.ac.uk#City">
  <owl:subClass ="http://jim.us#City"/>
</owl:Class>

<owl:Property rdf:about="http://ian.ac.uk#cv">
  <owl:subProperty rdf:resource="http://jim.us#Rsum"/>
</owl:Property>
```

can already be seen as an alignment expressing the equivalence of classes `Nappy` and `Diaper`, the coverage of class `City` in ontology identified by `ian` by `City` in that identified by `jim`, and that of the `cv` property by the `Rsum` property in the same ontologies. Moreover, any ontology, as soon as it involves entities from different ontologies, expresses alignments. For instance:

```
<owl:Class rdf:ID="Woman">
  <owl:equivalentClass>
    <owl:Class>
      <owl:subClassOf rdf:resource="http://www.example.org/ontology2#Person"/>
      <owl:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://www.example.org/ontology2#gender"/>
          <owl:hasValue rdf:resource="http://www.example.org/ontology2#W"/>
        </owl:Restriction>
      </owl:subClassOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

expresses that a `Woman` in ontology `http://www.example.org/ontology1` is equivalent to a `Person` in ontology `http://www.example.org/ontology2` whose `gender` is `W`.

47.1.1 Conclusion

Not surprisingly, the OWL language can be used as an alignment expression language. However, using it this way has some drawbacks:

1. It forces to use one ontology language: OWL. It is still possible to relate this way ontologies that are expressed in other languages (without benefiting from the construction of complex terms). However, the alignment will not benefit from the content of the ontologies themselves.
2. It mixes alignment and definitions. This is a problem for the clarity of alignments as well as for lightweight applications which do not want to interpret the OWL language.
3. It is interpreted only in the framework of a global interpretation of one OWL theory. It is difficult to use this expression for only importing data expressed under one ontology into another one (because this application requires sorting out definitions from correspondences).

Other languages have been set up for overcoming these problems: SKOS solves the first problem (but introduces its own language), SWRL solves problem (2) and C-OWL attempts to solve problem (3). These languages will be presented hereafter.

47.2 MAFRA Semantic bridging ontology (SBO)

MAFRA [da Silva, 2004; Mädche *et al.*, 2002] means "Mapping framework"[1]. It is a whole system for extracting mappings from ontologies and executing them as data transformation from one ontology to another. The system was first designed to work with the DAML+OIL language.

MAFRA does not define a real exchange format for ontology alignment. Rather, it provides an ontology, called Semantic Bridge Ontology. The instantiation of this ontology constitutes an ontology mapping document which is finally such a format. The serialisation of this format has not been described in detail in documents so we freely use our own transcription[2].

[1]It is also the name of a city in Portugal featuring a rich palace.

[2][Mädche *et al.*, 2002] presents SBO as a DAML+OIL ontology, but it seems to have evolved a lot since then and we have not been able to find a serialisation that could stand as a proper format.

The main concepts in this ontology are `SemanticBridge`s and `Service`s. A `SemanticBridge` is tied to the `Service`s that are able to implement the bridge (as a data transformation). A `Service` can be thought of as a function: $f : Arg^n \longrightarrow Arg^m$ which maps tuples of arguments into tuples of arguments. It is identified (one can imagine by a URI). The arguments are typed and can be ontology concepts, property paths, litterals or arrays of such. For instance, the service `CopyAttribute` (which copies an attribute from an ontology to another) is defined by:

```
pt.ipp.isep.gecad.mafra.services.ttransformations.CopyAttribute: path -> path
pt.ipp.isep.gecad.mafra.services.ttransformations.CountProperties: path -> integer
```

`SemanticBridges` (which can be `ConceptBridge`s or `PropertyBridge`s) express a relation between two sets of entities by composing elementary services that are applied to them. For instance, a `SemanticBridge` between two ontologies such as this one defines `Individual` and the target ontology defines both `Man` and `Woman` will be expressed in the following way:

```
ConceptBridge: Indiv2Woman
    x: <o1#Individual>; genre == 'W' -> <o2#Woman>
ConceptBridge: Indiv2Man
    x: <o1#Individual>; genre == 'M' -> <o2#Man>
exclusive: Indiv2Woman, Indiv2Man
```

Entities to be mapped are identified within the ontology (instances) through a path. Paths serve dual purposes of navigating within the ontology structure and providing the context further characterising the concerned entities. In this context paths play exactly the same role as in Xpath. They are further enriched with conditions (in the example above, `<o1#Individual>; genre == 'W'` is a path with condition that the final step `genre` has value `W` (the complete example involves regular expressions on the string value). More complex bridges can then be expressed, as an example given in [da Silva, 2004]:

```
PropertyBridge: spouseIn2noMarriages
    y: <o1#Individual:spouseIn>
->  <o2#Individual:noMarriages> = countProperties( y )
```

It means that the occurence of property `spouseIn` in ontology `o1` will be translated in the count of these properties to be recorded in the property `noMarriages` in ontology `o2`. Here `<o1#Individual:spouseIn>` is a path and `countProperties` is a service.

An ontology mapping document satisfying SBO is a collection of such bridges plus information on the concerned ontologies as well as constraints (such as the "exclusive" of the first example expressing that only one of the two bridges can be triggered).

47.2.1 Conclusion

SBO provided a framework for expressing alignments. This format is used as output of ontology matchers and input of data transformations.

The format provided by SBO is not very clear since all the language is described in UML. This is a minor problem that could be solved by exposing some RDF/XML format (a previous version of the framework had been described as a DAML ontology [Mädche *et al.*, 2002]). Moreover, this format is a relatively complex language that is tied to the MAFRA architecture.

It does not separate the declarative aspect of relations from the more operational one of service: the relations are described in functions of the services able to implement them. The services can be arbitrary small (like string concatenation) or large (like implementing a complete alignment by a program). On the one hand, this guarantees that these alignments can be used: SBO-documents can readily be used as data transformations. On the other hand, this does not help using these alignments in other ways (for merging ontologies or mediating queries for instance).

47.3 Contextualized OWL (C-OWL)

C-OWL is an extension of the OWL language to express mappings between heterogeneous ontologies. The constructs in C-OWL are called *bridge rules*, and they allow to express a family of semantic relations between concepts/roles and individuals interpreted in heterogeneous domains. Given two ontologies O_1 and O_2 a bridge rule from O_1 to O_2 expresses a semantic relation between a concept/role/individual of O_1 and a concept/role/individual of O_2. Bridge rules are directional in the sense that bridge rules from O_1 to O_2 are not the inverse of the bridge rules from O_2 to O_1.

Abstract syntax There are five types of bridge rules for concepts, five for roles and five for individuals. We report the abstract syntax with their intuitive in the following table.

Abstract syntax	Intuitive meaning
$i : A \xrightarrow{\sqsubseteq} j : B$	the concept A in ontology i is *more specific* that the concept B in the ontology j
$i : A \xrightarrow{\sqsupseteq} j : B$	the concept A in ontology i is *more general* that the concept B in the ontology j
$i : A \xrightarrow{\equiv} j : B$	the concept A in ontology i is *equivalent* to the concept B in the ontology j
$i : A \xrightarrow{\perp} j : B$	the concept A in ontology i is *disjoint* from the concept B in the ontology j
$i : A \xrightarrow{*} j : B$	the concept A in ontology i *overlaps* with the concept B in the ontology j

The above table reports only the bridge rules on concepts, but analogous expressions are possible in C-OWL when A and B are either roles or individuals.

Bridge rules form O_1 to O_2 must be read from the target ontology (O_2) viewpoint. Namely they express how the target ontology (ontology 2) sees or translates the source ontology (ontology 1) in itself. Furthermore, the relations "more general", "less general", etc., are not restricted to be mere set theoretical relations, as the domains of interpretation of the two ontologies may be completely different (a mediating relation between the elements of the domain of interpretation is required).

Concrete syntax C-OWL concrete syntax is given in XML form, and it looks like the one given in the following example of a C-OWL file

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
     <!ENTITY rdf   "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
     <!ENTITY rdfs  "http://www.w3.org/2000/01/rdf-schema#" >
     <!ENTITY xsd   "http://www.w3.org/2001/XMLSchema#" >
     <!ENTITY owl   "http://www.w3.org/2002/07/owl#" >
     <!ENTITY cowl  "http://www.itc.it/cowl#" >
   ]>
<rdf:RDF
  xmlns      ="&cowl;"
  xmlns:cowl ="&cowl;"
  xmlns:owl  ="&owl;"
  xmlns:rdf  ="&rdf;"
  xmlns:rdfs ="&rdfs;"
  xml:base   ="http://www.itc.it/cowl/example#"
  >
<cowl:Mapping>
    <cowl:sourceOntology>
      <owl:Ontology rdf:about="http://www.itc.it/ontologies/source.owl"/>
    </cowl:sourceOntology>
    <cowl:targetOntology>
      <owl:Ontology rdf:about="http://www.itc.it/ontologies/target.owl"/>
    </cowl:targetOntology>
    <cowl:bridgeRule>
      <cowl:Into>
        <cowl:source>
          <owl:Class rdf:about="http://www.itc.it/ontologies/source.owl#Article"/>
        </cowl:source>
        <cowl:target>
          <owl:Class rdf:about="http://www.itc.it/ontologies/target.owl#Publication"/>
        </cowl:target>
      </cowl:Into>
    </cowl:bridgeRule>
    <cowl:bridgeRule>
      <cowl:Onto>
        <cowl:source>
          <owl:Class rdf:about="http://www.itc.it/ontologies/source.owl#Person"/>
        </cowl:source>
        <cowl:target>
          <owl:Class rdf:about="http://www.itc.it/ontologies/target.owl#GraduateStudent"/>
        </cowl:target>
      </cowl:Onto>
    </cowl:bridgeRule>
    <cowl:bridgeRule>
      <cowl:Equivalent>
        <cowl:source>
          <owl:Class rdf:about="http://www.itc.it/ontologies/source.owl#Student"/>
        </cowl:source>
        <cowl:target>
          <owl:Class rdf:about="http://www.itc.it/ontologies/target.owl#Student"/>
        </cowl:target>
      </cowl:Equivalent>
    </cowl:bridgeRule>
    <cowl:bridgeRule>
</cowl:Mapping>
</rdf:RDF>
```

Semantics The full details of the formal semantic for C-OWL bridge rules is provided in [Bouquet *et al.*, 2003a]. The semantics of bridge rules from an ontology O_1 to ontology O_2 is given w.r.t. a distributed interpretation. A distributed interpretation for O_1 and O_2 is a 3-tuple $\mathfrak{I} = \langle \mathcal{I}_1, \mathcal{I}_2, r_{12} \rangle$ where $\mathcal{I}_1$ and $\mathcal{I}_2$ are models of O_1 and O_2 respectively, and r_{12}, called the *domain relation* is a subset of $\Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$. The domain relation models a translation function from the domain of interpretation of O_1 to the domain of interpretation of O_2. Intuitively $\langle d, d' \rangle \in r_{12}$ means that d' is one among the possible translation of d into $\Delta^{\mathcal{I}_2}$.

1. $\mathfrak{I} \models 1{:}\phi \xrightarrow{\sqsubseteq} 2{:}\psi$ if $r_{12}(\phi^{\mathcal{I}_i}) \subseteq \psi^{\mathcal{I}_j}$;
2. $\mathfrak{I} \models 1{:}\phi \xrightarrow{\sqsupseteq} 2{:}\psi$ if $r_{12}(\phi^{\mathcal{I}_i}) \supseteq \psi^{\mathcal{I}_j}$;
3. $\mathfrak{I} \models 1{:}\phi \xrightarrow{\equiv} 2{:}\psi$ if $r_{12}(\phi^{\mathcal{I}_i}) = \psi^{\mathcal{I}_j}$;
4. $\mathfrak{I} \models 1{:}\phi \xrightarrow{\perp} 2{:}\psi$ if $r_{12}(\phi^{\mathcal{I}_i}) \cap \psi^{\mathcal{I}_j} = \emptyset$;
5. $\mathfrak{I} \models 1{:}\phi \xrightarrow{*} 2{:}\psi$ if $r_{12}(\phi^{\mathcal{I}_i}) \cap \psi^{\mathcal{I}_j} \neq \emptyset$;

Decision procedure A tableaux based decision procedure for ontology spaces, i.e., a set of ontologies connected via bridge rules, is described in [Serafini *et al.*, 2005] and has been implemented in a peer-to-peer distributed reasoning system called DRAGO [3] and described in [Serafini and Tamilin, 2005]. The decision procedure supports only bridge rules between concepts, with the exception of the $\xrightarrow{*}$ bridge rules.

47.3.1 Conclusion

The C-OWL proposal can express relatively simple alignments (no constructed classes are expressed, only named classes are used). The more expressive part lays in the relations used by the mapping. These alignment have a clear semantics, however it is given from a particular standpoint: that of the target ontology. C-OWL is based on the OWL language but relatively independent from this language which is confined at expressing the entities (the alignment part being specific).

47.4 SWRL

Some people want to be able to express rules and not only concept definitions. SWRL [Horrocks *et al.*, 2004] (Semantic Web Rule Language) is a rule language for the semantic web. It extends OWL with an explicit notion of rule (from RuleML) that is interpreted as first order Horn-clauses. These rules can be understood as correspondences between ontologies (especially when elements from the head and the body are from different ontologies).

SWRL mixes the vocabulary from RuleML for exchanging rules with the OWL vocabulary for expressing knowledge. It defines a rule (`ruleml:imp`) with a body (`ruleml:body`) and head (`ruleml:head`) parts.

```
<ruleml:imp>
  <ruleml:_body>
    <swrlx:classAtom>
      <owlx:Class owlx:name="http://www.example.org/ontology2#Person" />
      <ruleml:var>p</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="http://www.example.org/ontology2#gender">
      <ruleml:var>p</ruleml:var>
      <owlx:Individual owlx:name="W" />
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:classAtom  swrlx:property="http://www.example.org/ontology1#Woman">
      <ruleml:var>p</ruleml:var>
    </swrlx:classAtom>
  </ruleml:_head>
```

[3] http://trinity.dit.unitn.it/drago

```
</ruleml:imp>
```

This last rule expresses that a `Person` in ontology `http://www.example.org/ontology2` with `W` as the value of its `gender` attribute is a `Woman` in ontology `http://www.example.org/ontology1`.

The introduction of variables within constructs of the OWL language provides more expressiveness to the language (in particular it allows to express what was called role-value maps in description logics or feature path equations in feature algebras). SWRL also provide a set of built-in predicates on the various datatypes provided by XML Schema as well as operators on collections (like count).

47.4.1 Conclusion

SWRL rules can be used for expressing the correspondences between ontologies. These correspondences are expressed between formulas and interpreted as Horn-clauses. They have the advantage over genuine OWL to be well identified as rules and are easier to manipulate as an alignment format than OWL which is also used to express ontologies.

Like the OWL example, these rules have the drawback of forcing the use of OWL and are interpreted as merging ontologies. Again, the expression of a rule like the one above fixes the use that can be made: the rule will help considering some people of the first ontology as women in the second ontology. However, the rules work as a set of logical rules, not rewrite rules, so this really provides the use for merging, not transforming[4].

47.5 Alignment format

As briefly sketched above and before [Euzenat, 2003], in first approximation, an alignment is a set of pairs of elements from each ontology. However, as already pointed out in [Noy and Musen, 2002a], this first definition does not cover all needs and all alignments generated. So [Euzenat, 2004] provided an Alignment format on several levels, which depends on more elaborate alignment definitions.

47.5.1 Alignment

The alignment description can be stated as follows:

a level used for characterising the type of correspondence (see below);

a set of correspondences which express the relation holding between entities of the first ontology and entities of the second ontology. This is considered in the following subsections;

an arity (default 1:1) Usual notations are 1:1, 1:m, n:1 or n:m. We prefer to note if the mapping is injective, surjective and total or partial on both side. We then end up with more alignment arities (noted with, 1 for injective and total, ? for injective, + for total and * for none and each sign concerning one mapping and its converse): ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:*. These assertions could be provided as input (or constraint) for the alignment algorithm or as a result by the same algorithm.

[4]http://co4.inrialpes.fr/align

This format is simpler than most of the alignment representations presented here, but is supposed producible by most matching tools.

To this strict definition can be added much more information (in particular when the format is expressed in RDF) such as:

– the generating algorithm;
– date of creation;
– is the alignment homogeneous (in language or entity).

47.5.2 Level 0

The very basic definition of a correspondence is the one of a pair of discrete entities in the language (identified by URIs). This first level of alignment has the advantage not to depend on a particular language. Its definition is roughly the following:

entity1 the first aligned entity. It is identified by an URI and corresponds to some discrete entity of the representation language.

entity2 the second aligned entity with the same constraint as entity1.

relation (default ”=”) the relation holding between the two entities. It is not restricted to the equivalence relation, but can be more sophisticated (e.g., subsumption, incompatibility [Giunchiglia and Shvaiko, 2003a], or even some fuzzy relation).

strength (default $\top$) denotes the confidence held in this correspondence. Since many alignment methods compute a strength of the relation between entities, this strength can be provided as a normalised measure. The measure should belong to an ordered set M including a maximum element $\top$ and a minimum element $\bot$. Currently, we restrict this value to be a float value between $0.$ and $1.$. If found useful, this could be generalised into any lattice domain.

id an identifier for the correspondence.

A simple pair can be characterised by the default relation ”=” and the default strength $\top$. These default values lead to consider the alignment as a simple set of pairs.

On this level, the aligned entities may be classes, properties or individuals. But they also can be any kind of complex term that is used by the target language. For instance, it can use the concatenation of firstname and lastname considered in [Rahm and Bernstein, 2001] if this is an entity, or it can use a path algebra like in:

```
hasSoftCopy.softCopyURI = hasURL
```

However, in the format described above and for the purpose of storing it in some RDF format, it is required that these entities (here, the paths) are discrete and identifiable by a URI.

A full example of the Level 0 Alignment format in RDF is the following:

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<!DOCTYPE rdf:RDF SYSTEM "align.dtd">

<rdf:RDF xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/alignment'
         xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
<Alignment>
```

```
  <xml>yes</xml>
  <level>0</level>
  <type>**</type>
  <onto1>http://www.example.org/ontology1</onto1>
  <onto2>http://www.example.org/ontology2</onto2>
  <map>
    <Cell>
      <entity1 rdf:resource='http://www.example.org/ontology1#reviewedarticle'/>
      <entity2 rdf:resource='http://www.example.org/ontology2#article'/>
      <measure rdf:datatype='&xsd;float'>0.6363636363636364</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='http://www.example.org/ontology1#journalarticle'/>
      <entity2 rdf:resource='http://www.example.org/ontology2#journalarticle'/>
      <measure rdf:datatype='&xsd;float'>1.0</measure>
      <relation>=</relation>
    </Cell>
  </map>
</Alignment>
</rdf:RDF>
```

It describes a many-to-many Level 0 alignment between two bibliographic ontologies. It contains two correspondences that identify `reviewedarticle` to `article` and `journalarticle` to `journalarticle respectively`. These correspondence use the equivalence relation and a confidence measure (.64 in the former case and 1. in the latter).

Level 0 alignments are basic but found everywhere: there are no algorithm that cannot account for such alignments. It is, however, somewhat limited: there are other aspects of alignments that can be added to this first approximation.

47.5.3 Level 1

Level 1 replaces pairs of entities by pairs of sets (or lists) of entities. A level 1 correspondence is thus a slight refinement of level 0, which fills the gap between level 0 and level 2. However, it can be easily parsed and is still language independent.

47.5.4 Level 2 (L)

More general correspondence expressions can be useful. For instance, [Masolo *et al.*, 2003] provides a number of bridges from their ontology of services to the currently existing semantic web service description language in first order logic. These kind of correspondences can be expressed with level 2 alignments.

Level 2 considers sets of expressions of a particular language (L) with variables in these expressions. Correspondences are thus directional and correspond to a clause:

$$\forall \overline{x_f}(f \implies \exists \overline{x_g} g)$$

in which the variables of the left hand side are universally quantified over the whole formula and those of the right hand side (which do not occur in the left hand side) are existentially quantified.

This level can express correspondences like:

$$\forall x, z\; grandparent(x, z) \implies \exists y; parent(x, y) \wedge parent(y, z)$$

This kind of rules (or restrictions) is commonly used in logic-based languages or in the database world for defining the views in "global-as-view" of "local-as-view" approaches [Calvanese *et al.*, 2002b]. It also resembles the SWRL rule language [Horrocks *et al.*, 2004] when used with OWL (see §49.1.3 for a simple example of such rules). These rules can also be generalised to any relation and drop the orientation constraint.

Level 2 can be applied to other languages than OWL (SQL, regular expressions, F-Logic, etc.). For instance, the expression can apply to character strings and the alignment can denote concatenation like in:

```
name = firstname+" "+lastname
```

The Alignment format has been given an OWL ontology and a DTD for validating it in RDF/XML. It can be manipulated through the Alignment API which is presented below.

47.5.5 Alignment API

A JAVA API can be used for implementing this format and linking to alignment algorithms and do2002a procedures. It is briefly sketched here.

Classes

The OWL API is extended with the (`org.semanticweb.owl.align`) package which describes the Alignment API. This package name is used for historical reasons. In fact, the API itself is fully independent from OWL or the OWL API.

It is essentially made of three interfaces. We present here, under the term "features", the information that the API implementation must provide. For each feature, there are the usual reader and writer accessors:

`Alignment` The Alignment interface describes a particular alignment. It contains a specification of the alignment and a list of cells. Its features are the following:

- **`xml`** (value: "yes"/"no") indicates if the alignment can be read as an XML file compliant with the DTD;
- **`level`** (values "0", "1", "2*") indicates the level of alignment format used;
- **`type`** (values: "11", "1?", "1+", "1*", "?1", "??", "?+", "?*", "+1", "+?", "++", "+*", "*1", "*?", "?+", "**") the type of alignment;
- **`onto1`** (value: URL) the first ontology to be aligned;
- **`onto2`** (value: URL) the second ontology to be aligned;
- **`map`** (value: `Cell`*) the set of correspondences between entities of the ontologies.

`Cell` The Cell interface describes a particular correspondence between entities. It provides the following features:

- **`rdf:resource`** (value: URI) the URI identifying the current correspondence;
- **`entity1`** (value: URI) the URI of some entity of the first ontology;

- **`entity2`** (value: URI) the URI of some entity of the second ontology;
- **`measure`** (value: float between 0. and 1.) the confidence in the assertion that the relation holds between the first and the second entity (the higher the value, the higher the confidence);
- **`relation`** (value: `Relation`) the relation holding between the first and second entity.

`Relation` The Relation interface does not mandate any particular feature.

To these interfaces, implementing the format, are added a couple of other interfaces:

`AlignmentProcess` The AlignmentProcess interface extends the Alignment interface by providing an `align` method. This interface must be implemented for each alignment algorithm.

`Evaluator` The Evaluator interface describes the comparison of two alignments (the first one could serve as a reference). Its features are the following:

- **`align1`** (value: URI) a first alignment, sometimes the reference alignment;
- **`align2`** (value: URI) a second alignment which will be compared with the first one.

An additional `AlignmentException` class specifies the kind of exceptions that are raised by alignment algorithms and can be used by alignment implementations.

Functions

Of course, this API does provide support for manipulating alignments. It offers a number of services for manipulating the API. As in [Bechhofer *et al.*, 2003], these functions are separated in their implementation. The following primitives are available:

parsing/serialising an alignment from a file in RDF/XML (`AlignmentParser.read()`, `Alignment.write()`);

computing the alignment, with input alignment (`Alignment.align(Alignment, Parameters)`);

thresholding an alignment with threshold as argument (`Alignment.cut(double)`);

hardening an alignment by considering that all correspondences whose strength is strictly greater than the argument is converted to $\top$, the others being $\bot$ (`Alignment.harden(double)`);

comparing one alignment with another (`Evaluator.eval(Parameters)`) and serialising them (`Evaluator.write()`);

outputting alignment in a particular format (SWRL, OWL, XSLT, RDF, etc.) (`Alignment.render(visitor)`);

In addition, alignment and do2002a algorithms accept parameters. These are put in a structure that allows storing and retrieving them. The parameter name is a string and its value is any Java object. It is advised to have them serialised as a string because external interface (such as the Procalign command-line interface will provide them as such). The parameters can be the various weights used by some algorithms, some intermediate thresholds or the tolerance of some iterative algorithms.

This Alignment API has been implemented in Java on top of the OWL API. This implementation has been used for various purposes: online alignment at Innsbruck, Evaluation tool in the Ontology Alignment Evaluation Initiative, many extensions of it use it for implementing matching algorithms (oMAP from CNR, OLA form INRIA/University of Montral) or support it (FOAM from Karlsruhe, CMS from Southampton).

47.5.6 Conclusion

The Alignment format used with the Alignment API allows to express alignments without committing to some language. It is not targeted towards a particular use of the alignments and offer generators for a number of other formats. But, by opposition to the languages presented so far, this genuine format does not offer much expressiveness.

However, one good feature of this format is its openness which allows to introduce new relations and if necessary new type of expressions while keeping the compatibility with poorly expressive languages.

47.6 SEKT Mapping language (OML)

47.6.1 Introduction

The Ontology Management Working Group[5], jointly with the SEKT[6] and DIP[7] European projects, aims at developping a complete ontology management suite for the semantic web. As part of this effort the SEKT project has developed an ontology mapping language [de Bruijn *et al.*, 2004]. This language provides a complete format as a basis to represent ontology mappings. This format serves to express mappings resulting from a matching algorithm or from a graphical mapping tool. It has the advantage of being independant from the ontology language, thus giving a common basis to research on schema matching techniques.

The Ontology Mapping Language is also used to specify semantic web services data mediators in the WSMO (Web Services Modeling Ontology)[8] [Roman *et al.*, 2004]. The mediation between semantic web services is a crucial part in this field. When different services have to communicate together but are described using different ontologies, a data mediator comes into play to align the different vocabularies. The core part of a mediator definition is a mapping between the two ontologies plus some non functional properties. At run-time, the mediator is used to rewrite queries and transform instances, thus allowing communication between the services.

This section presents the language in its human-readable abstract syntax form. An RDF vocabulary is also available [Scharffe, 2005].

47.6.2 Presentation of the mapping language

This language provides a set of constructs to express mappings between classes, attributes, relations and instances of an ontology.

A set of operators associated to each type of entities gives the possibility to combine them. Table 47.2 presents the different operators for each type of entity.

Each operator has a cardinality, an effect and some related semantics. The semantics are related to the logical formalism used to represent the ontologies. For example the semantics of the 'and' operator between two classes is linked to the semantic of 'and' in OWL if the mappings are grounded to OWL.

The conditions under which the mappings are valid are specified by introducing a conditional field in the mapping rules. These conditions may be a class condition or an attribute condition. The

[5]http://www.omwg.org
[6]http://www.sekt-project.org
[7]http://dip.semanticweb.org
[8]http://www.wsmo.org

Language Construct	Description
ClassMapping	Mapping between two classes
AttributeMapping	Mapping between two attributes
RelationMapping	Mapping between two relations
ClassAttributeMapping	Mapping between a class and an attribute
ClassRelationMapping	Mapping between a class and a relation
ClassInstanceMapping	Mapping between a class and an instance
IndividualMapping	Mapping between two instances

Table 47.1: SEKT-ML mapping types.

Entity	Operator
Class	and, or, not, join
Attribute	and, or, not, inverse, symetric, reflexive, transitive closure, join
Relation	and, or, not, join

Table 47.2: SEKT-ML logical constructions.

class conditions are based on their nested attribute values, occurrences or types while the attribute condition are based on their own values or types. Table 47.3 displays the different conditions the mapping language can express.

To get a clear but expressive language, limited constructs for the most common cases of mappings are defined, allowing the user to define arbitrary logical expressions to represent those which do not have constructs. These logical expressions must be written according to the two ontology modelling languages.

The syntax of this language has been designed to be intuitive and human readable. This results in a verbose syntax far from the often used XML syntaxes. However XML and RDF syntaxes are available. The language comes with a Java API that provides parsing and serialising methods to and from an object model of the mapping document. Figure 47.1 presents two simple ontologies representing the same domain but with a different modelling perspective. We will give the mapping

Range	Name
Class conditions	attributeValueCondition
	attributeTypeCondition
	attributeOccurenceCondition
Attribute Conditions	valueCondition
	typeCondition

Table 47.3: SEKT-ML attribute comparators

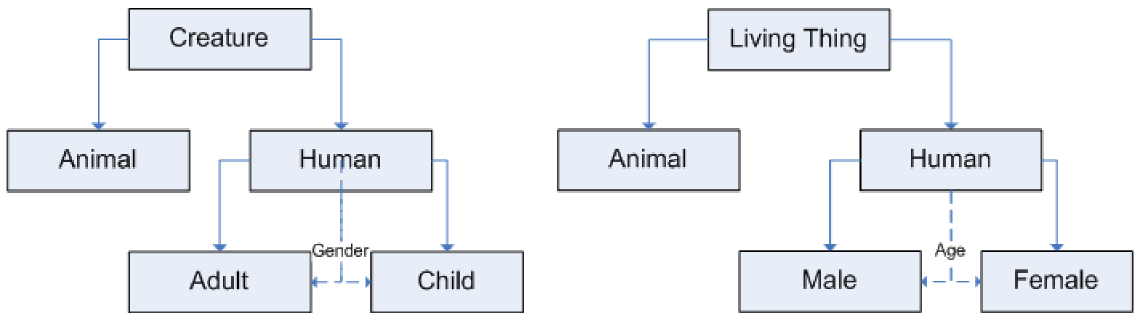


Figure 47.1: Two ontologies.

having the 'Living Thing' ontology as source and the 'Creature' ontology as target.

The top concepts `Living_Thing` and `Creature` presents a terminological mismatch of synonymy. On both ontologies the concepts `human` and `animal` are modelled using the same label. These three cases are simple class to class mappings expressed in the mapping language. Here are the statements representing these mappings.

```
classMapping(
  annotation(<"rdfs:label">  'Creature to LivingThing')
  annotation(<"http://purl.org/dc/elements/1.1/description">
    'Map the person concept to the livingThing concept')
  bidirectional
  <"http://ontologies.omwg.org/creature#Creature">
  <"http://ontologies.omwg.org/livingThing#Living Thing">)

classMapping(
  annotation(<"rdfs:label"> 'Animal to Animal')
  bidirectional
  <"http://ontologies.omwg.org/creature#Animal">
  <"http://ontologies.omwg.org/livingThing#Animal">)

classMapping(
  annotation(<"rdfs:label"> 'human to human')
  bidirectional
  <"http://ontologies.omwg.org/creature#Human">
  <"http://ontologies.omwg.org/livingThing#Human">)
```

The annotation fields allow the input of annotations, for instance, title or description. This field is also used when the mappings are resulting from an algorithm, whereby information like the confidence degree of the mapping and the algorithm used are stated. The RDFS and Dublin Core namespaces are used to indicate the nature of the descriptions. Another field express the directionality of the mappings. By default a mapping is bidirectional, meaning that the source and target entities are equivalent. It may also be unidirectional, meaning that the target entity somehow subsumes the source one.

The complexities come when mapping the `Male` and `Female` concepts in the `living Thing` ontology to the subconcepts of `Human`, namely `Adult` and `Child` in the `creature` ontology. A `Human` `Male` or `Female` is an `Adult/Child` if his or her `age` is greater than or equal to/lower than 18. This kind of mapping is represented using a condition. The concepts are considered mapped only if the condition specified in the mapping rule is valid. Following is the representation of such a condition for this example.

```
classMapping(
  annotation(<"rdfs:label"> 'conditional female to adult')
  unidirectional
  <"http://ontologies.omwg.org/creature#Female">
  <"http://ontologies.omwg.org/livingThing#Adult">
  attributeValuecondition(
     <"http://ontologies.omwg.org/Creature#age '>=18'))

classMapping(
  annotation(<"rdfs:label"> 'conditional female to child')
  unidirectional
  <"http://ontologies.omwg.org/creature#Female">
  <"http://ontologies.omwg.org/livingThing#Child">
  attributeValuecondition(
       <"http://ontologies.omwg.org/creature#age '<18'))
```

The same rules must then be written for the `male` concept in order to realise a complete mapping. A mapping between the `female`/`male` concepts in the source ontology and the `female`/`male gender` attribute in the target one may also be created. This kind of mapping is saying: ”The instances of the female concept in the source ontology are equivalent to the instances having a gender attribute with the value ’female’ in the target ontology”. Here is the representation in terms of the mapping language.

```
classAttributeMapping(
  annotation(<"rdfs:label"> 'map female to gender:female')
  unidirectional
  <"http://ontologies.omwg.org/creature#Female">
  <"http://ontologies.omwg.org/livingThing#gender:female">)

classAttributeMapping(
  annotation(<"rdfs:label"> 'map the male to the gender:male')
  unidirectional
  <"http://ontologies.omwg.org/creature#Male">
  <"http://ontologies.omwg.org/livingThing#gender:male">)
```

47.6.3 Conclusion

The SEKT Mapping language is an expressive alignment format offering many kinds of relations and entity constructor to the users. One of its main advantages is its ontology language independence, giving a common format for expressing mappings.

So this proposal has a middle man position: it is independent from any particular language but expressive enough for covering a large part of the other languages.

47.7 SKOS

SKOS [Miles and Brickley, 2005b; Miles and Brickley, 2005a] means “Simple Knowledge Organisation System”. The SKOS core vocabulary is an RDF Schema aiming at expressing relationships between lightweight ontologies (as known as folkosomies) or thesauri. It is currently under development and thus quite unstable at the time of writing.

The goal of SKOS is to be a layer on top of other formalisms able to express the links between entities in these formalisms.

Concept and relation descriptions

SKOS allows to identify the concepts that are present in the other ontologies. The concept description part of SKOS, seems quite redundant with other languages of that family; it seems that it is designed for being able to take advantage of these concepts (e.g., in a GUI) rather to only express the alignments.

Here are such concept descriptions dedicated to the description of the SKOS and OWL concepts. It defines various ways of presenting the concept (with labels in several languages and alternate labels and symbols). It also provides the opportunity to add various notes and informal definitions to the concept.

```
<skos:Concept rdf:about="http://www.w3c.org#skos">
  <skos:prefLabel>Simple Knowledge Organisation System</skos:prefLabel>
  <skos:altLabel>SKOS</skos:altLabel>
  <skos:altLabel xml:lang="fr">Systme simple d'organisation de la
                   connaissance</skos:altLabel>
  <skos:hiddenLabel xml:lang="fr">SSOC</skos:hiddenLabel>
  <skos:definition>The SKOS core vocabulary is a RDFS schema which aims at
                   expressing relationships between lightweight ontologies
                   (as known as folkosomies) or thesauri.</skos:definition>
  <skos:editorialNote>This is not an official definition</skos:editorialNote>
</skos:Concept>

<skos:Concept rdf:about="http://www.w3c.org#owl">
  <skos:prefLabel>Web ontology language</skos:prefLabel>
  <skos:altLabel>OWL</skos:altLabel>
  <skos:altLabel xml:lang="de">EULE</skos:altLabel>
  <skos:prefSymbol rdf:resource="http://www.cs.umd.edu/users/hendler/2003/OWL2.gif" />
  <skos:definition>OWL is an ontology language for the web recommended by W3C.</skos:definition
  <skos:historyNote>OWL is not an acronym</skos:historyNote>
  <skos:hasTopConcept rdf:resource="&owl;#Thing"/>
</skos:Concept>
```

SKOS also alows to describe collections of concepts given by their enumeration:

```
<skos:Collection rdf:about="http://www.example.org#alignmentFormats">
  <rdfs:label>Alignment formats</rdfs:label>
  <skos:member rdf:resource="http://www.w3c.org#skos"/>
  <skos:member rdf:resource="http://www.w3c.org#owl"/>
  <skos:member rdf:resource="http://www.w3c.org#swrl"/>
  <skos:member rdf:resource="http://www.wsmg.org#ml"/>
  <skos:member rdf:resource="http://knowledgeweb.semanticweb.org/heterogeneity#alignapi"/>
  <skos:member rdf:resource="http://www.example.org#mafra"/>
</skos:Collection>
```

Concept relations

SKOS defines so-called “semantic relationships” that express relations between SKOS concepts. For instance, that a term used in a thesauri is broader than another. There are three such relations as defined in Table 47.4.

property	domain	range	inverse	property
broader	concept	concept	narrower	transitive
related	concept	concept	related	symmetric

Table 47.4: SKOS relation properties.

property	domain	range	inverse
subject	resources	concept	isSubjectOf
primarySubject	resource	concept	isPrimarySubjectOf

Table 47.5: SKOS annotation properties.

The relations between concepts enable the assertion of the relative inclusion of concepts as broader or narrower terms as well as another, informal, relation. The following displays the `RDFSchema` concept that is narrower than `ontologyLanguage` but broader than `SKOS`. It is also related to `folkosomyLanguages`.

```
<skos:Concept rdf:about="http://www.w3c.org#rdfschema">
  <skos:prefLabel>RDF Schema</skos:prefLabel>
  <skos:altLabel>RDFS</skos:altLabel>
  <skos:broader rdf:resource="http://www.example.org#ontologyLanguage"/>
  <skos:narrower rdf:resource="http://www.w3c.org#skos"/>
  <skos:related rdf:resource="http://www.example.org#folkosomyLanguage"/>
  <skos:editorialNote>This is not an official definition</skos:editorialNote>
<skos:Concept/>
```

Broader and narrower are transitive properties while related is symmetric.

Annotations

In addition, but of least interest here, SKOS defines annotation properties that enable users to use SKOS concepts for describing resources (annotate them directly with SKOS). It defines the vocabulary displayed in Table 47.5.

This is used below to express that `SKOS` is the `primarySubjectOf` its specification and a `subjectOf` this document.

```
<skos:Concept rdf:about="http://www.w3c.org#skos">
  <skos:isPrimarySubjectOf rdf:resource="http://www.w3.org/TR/2005/swbp-skos-core-spec" />
  <skos:isSubjectOf rdf:resource="http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/d
</skod:Concept>
```

and the SKOS guide [Miles and Brickley, 2005a] could have the following annotations:

```
<foaf:Document>
  <skos:primarySubject rdf:resource="http://www.w3c.org#skos" />
  <skos:subject rdf:resource="http://www.w3c.org#rdfs" />
</foaf:Document>
```

meaning that its `primarySubject` is `SKOS` and a `secondarySubject` is `RDFSchema`.

47.7.1 Conclusion

SKOS has the advantage of being a lightweight vocabulary defining from the ground a rich collection of relations between entities. Since it uses URIs for referring to objects it is fully integrated in the semantic web architecture and is not committed to a particular language. In fact one of the main advantage of SKOS is that it lifts any kind of organised description into an easily usable set of classes. The relation part has the advantage of being very general but the drawback of lacking formal semantics (more semantics on these terms can be brought by using the OWL vocabulary).

One of the main interest of SKOS, would be to define relations and concepts as instantiating the SKOS vocabulary like in:

```
<rdf:Class rdf:about="http://www.w3c.org/owl#Class">
  <rdfs:subClassOf rdf:resource="http://www.w3c.org/skos#Concept"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.w3c.org/owl#subClassOf">
  <rdfs:subPropertyOf rdf:resource="http://www.w3c.org/skos#narrower"/>
</rdf:Property>
```

However, this mixes three vocabularies (RDFS, OWL and SKOS). This is one of the main weakness of SKOS. Like other formats which do not separate the ontologies from the correspondences, SKOS, in its most convenient form mixes the highest power of RDF Schema and the expression of the alignments. This form of extensibility (through RDFS) prevents any non RDFS understanding application to fully grasp SKOS content.

Chapter 48

Comparison of existing formats

We have drawn local conclusions concerning each format presented in the previous chapter. We will here compare these formats globally in order to see emerging patterns. For that purpose, we define a set of do2002a criteria (§48.1), we apply them to the different formats (§48.2) and we finally analyse these results (§48.3).

48.1 Criteria

In order to meet the requirements stated in Chapter **??**, we explicit them here in a number of criteria to be applied to the existing systems. Several kinds of criterion can be presented with regard to each of the requirements.

48.1.1 Web compatibility

The capacity of the format to be manipulated on the web. This involves, its possible expression in XML, RDF and/or RDF/XML, as well as, the possibility to identify entities by URIs. This should, in principle, enable the extensibility of the format by introducing new properties as well as the referencing of particular correspondences individually.

This aspect is covered by the RDF/XML and URI criteria of Table 48.1.

48.1.2 Languages independence

The ability to express alignments between entities described in different languages. This is often related to the use of URIs. In fact, language independence is mostly related to simplicity.

This aspect is covered by the Language and Model criteria of Table 48.1.

48.1.3 Simplicity

The capacity to be dealt with in a simple form by simple tools. In particular, requiring inference for correctly manipulating the alignment (or requiring that the alignment format covers an important part of some ontology representation language), is not a sign of simplicity. A well structured format will help achieving this goal.

This aspect is covered by the Relations, Terms, Type rest, Cardinality, Variables and Built-in criteria of Table 48.1.

$$\begin{tikzcd}
\texttt{speed} \arrow[r, leftrightarrow, "="] & \texttt{velocity} \\
mph \arrow[r, shift left, "\lambda x.x \times 0.447"] & m/s \arrow[l, shift left, "\lambda x.x \times 2.237"]
\end{tikzcd}$$

Figure 48.1: Two conceptually equivalent properties, with different associated mappings.

48.1.4 Expressiveness

The capacity of the format to express complex alignments. This means that alignments are not restricted to matching entities identifies by URIs but can assemble them. The constructions for expressing alignments can be arbitrary complex (in fact, it can be more complex than the knowledge expressed in the ontologies).

This aspect is covered by the Relations, Terms, Type rest, Cardinality, Variables and Built-in criteria of Table 48.1.

This expressiveness can be considered from the standpoint of the richness of constructors available for expressing the terms in correspondence and the relations that can be expressed between these terms. In particular, we can distinguish the kind of terms, typing constraints, cardinality constraints, interdependence constraints expressed through variables and paths.

48.1.5 Extendibility

Extendibility is the capacity to extend the format with specific purpose information in such a way that the tools which use this format are not disturbed by the extensions. Most of the system presented here exhibit one kind of extendibility tied to the use of RDF which allows any new relation and object to be added. More particular kind of extendibility are exhibited by SBO and SEKT Mapping language in the plug-in architechture enabling the addition of new transformations, and the Alignment format in which the language itself can be extended.

This aspect is covered by the "+" signs in Table 48.1.

48.1.6 Purpose independence

Purpose independence is rather a consequence of various factors expressed below. We mention the intended use of each of the formats. It is clear for instance that a format designed for data integration, with very precise selection constraints, will rather be difficult to use in transforming ontologies. As an example, consider an ontology pair with a couple of equivalent attributes `speed` and `velocity` as in Table 48.1. A schema-level purpose independent alignment would record that these two properties are equivalent. However, when using this alignment, it may be necessary to use more information, namely that the first one is expressed in miles-per-hour and the second one is expressed in meter-per-second, so equivalent values require conversions.

This aspect is covered by the Target application criterion of Table 48.1.

48.1.7 Executability

Executability is the capacity to be directly usable in mediators. This means there are tools available for directly interpreting the format as a program processing knowledge. Executability is rather

Format	OWL	SBO	C-OWL	SWRL	Alignment	SEKT-ML	SKOS
Target app.	Merging	Data transf.	Data int.	Data int.	Generic	Data transf.	Merging
Language	OWL	UML	OWL	OWL	+		RDFS
Model	OWL		OWL+	OWL			
Execution	Logical	Transf	Logical	Logical		Logical	Alg.
RDF/XML	√		√	√	√	√	√
URI	√		√	√	√	√	√
Measures						√	
Relations	sc/sp		sc/sp	imp	sc/sp+	sc/sp/...	sc/sp
Terms	C/P/I	C/P/I	C/P/I	C/P/I	URI	C/P/I	C/P
Type rest	√	√	√	√		√	
Cardinality	√		√	√			
Variables				√			
Built-in		√+		√	+		

Table 48.1: Features of the presented systems

opposed to language independence.

This aspect is covered by the Execution criterion of Table 48.1.

48.2 Comparison

Table 48.1 provides the value of all the formats presented in the previous section for each of the criterion.

48.3 Synthesis

In fact, the real difference between these formats lies in the continuum between:

– very general languages easy to understand but unable to express the complexity of complex alignments (SKOS, Level 0 Alignment format), and
– very expressive languages which semantics dictates the use and which requires deep understanding of the language (OWL, SWRL, C-OWL, MAFRA).

The SEKT Mapping language stands in the middle of this continuum.

While most of the matching algorithms are only able to express the first kind of alignments, both kinds of languages are useful.

Most of the expressive formats have a surface heterogeneity due to the languages on which they are based (UML, OWL, WSML), however, they have very similar features for refering to ontology constructs (classes, properties), using logical formula constructions (conjunction, implications, quantifiers), as well as datatype and collection buit-in operators. It is even surprising that there is not much heterogeneity in these expressive languages given that complexity is a factor of: the language used for expressing the ontologies, the language used for expressing the related entities, the semantics given to alignments and the language used for expressing relations.

It would thus be useful to avoid that these expressing languages to commit to one kind of language so that their results cannot be used by others. We investigate hereafter, how this can be achieved.

Chapter 49

Interoperability

This chapter considers solutions that can help reaching short term convergence. It should help potential users who hesitate to commit to a particular format to be able to switch as much as possible from one format to another.

To that extent, there are different possible solutions. The first one involves using some transformation language (like XSLT) for translating from one format to another. This requires (and shows the benefit) of having the format in XML syntax.

Another solution consists of benefiting from API to generate the adequate format. This opportunity is mostly provided by language independent formats (the Alignment format and SEKT Mapping language).

We present below a number of these bridges between languages, first from the Alignment API (Section 49.1 and second form the SEKT Mapping language (Section 49.2.

49.1 Generating SEKT-ML/C-OWL/SKOS from Alignment API

The obtained alignment can, of course, be generated in the RDF serialisation form of the Alignment format. However, there are other formats available.

The API provides the notion of a visitor of the alignment cells. These visitors are used in the implementation for rendering the alignments. So far, the implementation is provided with four such visitors:

RDFRendererVisitor displays the alignment in the RDF format described in §47.5. An XSLT stylesheet is available for displaying the alignments in HTML from the RDF/XML format.

OWLAxiomsRendererVisitor generates an ontology merging both aligned ontologies and comprising OWL axioms for expressing the subsumption, equivalence and exclusivity relations.

XSLTRendererVisitor generates an XSLT stylesheet for transforming data expressed in the first ontology in data expressed in the second ontology;

COWLMappingRendererVisitor generates a C-OWL mapping [Bouquet and Serafini, 2003], i.e., a set of relations expressed between elements (in fact classes) of two ontologies.

SWRLRendererVisitor generates a set of SWRL [Horrocks *et al.*, 2004] rules for inferring from data expressed in the first ontology the corresponding data with regard of the second ontology.

SEKTMappingRendererVisitor generates a mapping document as was defined in the SEKT document [de Bruijn *et al.*, 2004].

SKOSRendererVisitor generates a SKOS mapping document.

Some of these methods, like XSLT or SWRL, take the first ontology in the alignment as the source ontology and the second one as the target ontology.

49.1.1 Generating axioms

OWL itself provides tools for expressing axioms corresponding to some relations that we are able to generate such as subsumption (`subClassOf`) or equivalence (`equivalentClass`). From an alignment, the `OWLAxiomsRendererVisitor` visitor generates an ontology that merges the previous ontologies and adds the bridging axioms corresponding to the cells of the alignment.

They can be generated from the following command-line invocation:

```
$ java -jar lib/procalign.jar -i fr.inrialpes.exmo.align.impl.SubsDistNameAlignment
   file://localhost$CWD/rdf/onto1.owl file://localhost$CWD/rdf/onto2.owl -t .6
   -r fr.inrialpes.exmo.align.impl.OWLAxiomsRendererVisitor
```

which returns:

```
<rdf:RDF
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <owl:Ontology rdf:about="">
    <rdfs:comment>Aligned ontollogies</rdfs:comment>
    <owl:imports rdf:resource="http://www.example.org/ontology1"/>
    <owl:imports rdf:resource="http://www.example.org/ontology2"/>
  </owl:Ontology>

  <owl:Class rdf:about="http://www.example.org/ontology1#reviewedarticle">
    <owl:equivalentClass rdf:resource="http://www.example.org/ontology2#article"/>
  </owl:Class>

  <owl:Class rdf:about="http://www.example.org/ontology1#journalarticle">
    <owl:equivalentClass rdf:resource="http://www.example.org/ontology2#journalarticle"/>
  </owl:Class>

</rdf:RDF>
```

49.1.2 Generating translations

Alignments can be used for translation as well as for merging. Such a transformation can be made on a very syntactic level. The most neutral solution seems to generate translators in XSLT. However, because it lacks deductive capabilities, this solution is only suited for transforming data (i.e., individual descriptions) appearing in a regular form.

The `XSLTRendererVisitor` generates transformations that recursively replace the names of classes and properties in individuals. The renderer produces stylesheets like:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
```

```
  <xsl:template match="http://www.example.org/ontology1#reviewedarticle">
    <xsl:element name="http://www.example.org/ontology2#article">
      <xsl:apply-templates select="*|@*|text()"/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="http://www.example.org/ontology1#journalarticle">
    <xsl:element name="http://www.example.org/ontology2#journalarticle">
      <xsl:apply-templates select="*|@*|text()"/>
    </xsl:element>
  </xsl:template>

  <!-- Copying the root -->
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>

  <!-- Copying all elements and attributes -->
  <xsl:template match="*|@*|text()">
    <xsl:copy>
      <xsl:apply-templates select="*|@*|text()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

49.1.3 Generating SWRL Rules

Finally, this transformation can be implemented as a set of rules which will “interpret” the correspondence. This is more adapted than XSLT stylesheets because, we can assume that a rule engine will work semantically (i.e., it achieves some degree of completeness with regard to the semantics) rather than purely syntactically.

The `SWRLRendererVisitor` transforms the alignment into a set of SWRL rules which have been defined in [Horrocks *et al.*, 2004]. The result on the same example will be the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<swrlx:Ontology swrlx:name="generatedAl"
                xmlns:swrlx="http://www.w3.org/2003/11/swrlx#"
                xmlns:owlx="http://www.w3.org/2003/05/owl-xml"
                xmlns:ruleml="http://www.w3.org/2003/11/ruleml#">
  <owlx:Imports rdf:resource="http://www.example.org/ontology1"/>

  <ruleml:imp>
    <ruleml:_body>
      <swrlx:classAtom>
        <owlx:Class owlx:name="http://www.example.org/ontology1#reviewedarticle"/>
        <ruleml:var>x</ruleml:var>
      </swrlx:classAtom>
    </ruleml:_body>
    <ruleml:_head>
      <swrlx:classAtom>
        <owlx:Class owlx:name="http://www.example.org/ontology2#journalarticle"/>
        <ruleml:var>x</ruleml:var>
      </swrlx:classAtom>
    </ruleml:_head>
  </ruleml:imp>

...
</swrlx:Ontology>
```

Of course, level 2 alignments would require specific renderers targeted at their particular languages.

49.1.4 Generating C-OWL mappings

The `COWLMappingRendererVisitor` transforms the alignment into a set of C-OWL mapping which have been defined in [Bouquet and Serafini, 2003]. The result on the same example will be the following:

```
<rdf:RDF
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:cowl="http://www.itc.it/cowl#"
    xml:base="http://www.itc.it/cowl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

  <cowl:Mapping rdf:ID="">
    <cowl:sourceOntology>
      <owl:Ontology rdf:about="http://www.example.org/ontology1"/>
    </cowl:sourceOntology>
    <cowl:targetOntology>
      <owl:Ontology rdf:about="http://www.example.org/ontology2"/>
    </cowl:targetOntology>
    <cowl:bridgeRule>
     <cowl:Equivalent>
       <cowl:source>
         <owl:Class rdf:about="http://www.example.org/ontology1#reviewedarticle"/>
       </cowl:source>
       <cowl:target>
         <owl:Class rdf:about="http://www.example.org/ontology2#journalarticle"/>
       </cowl:target>
     </cowl:Equivalent>
    </cowl:bridgeRule>
  ...
  </cowl:Mapping>
</rdf:RDF>
```

49.1.5 Generating SEKT-ML mappings

The `SEKTMappingRendererVisitor` transforms the alignment into a SEKT mapping document which have been defined in [de Bruijn *et al.*, 2004]. The result on the same example is the following:

```
MappingDocument(<"">
  source(<"http://www.example.org/ontology1">)
  target(<"http://www.example.org/ontology2">)
  classMapping( <"#s44261">
    bidirectional
    <"http://www.example.org/ontology1#reviewedarticle">
    <"http://www.example.org/ontology2#article">
  )

  classMapping( <"#s4201">
    bidirectional
    <"http://www.example.org/ontology1#journalarticle">
    <"http://www.example.org/ontology2#journalarticle">
  )

)
```

Of course, level 2 alignments would require specific renderers targeted at their particular languages.

49.1.6 Generating SKOS

The `SKOSRendererVisitor` transforms the alignment into a SKOS document which have been defined in [de Bruijn *et al.*, 2004]. The result on the same example is the following:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#">

  <skos:Concept rdf:about="http://www.example.org/ontology1#journalarticle">
    <skos:related rdf:resource="http://www.example.org/ontology2#journalarticle"/>
  </skos:Concept>

  <skos:Concept rdf:about="http://www.example.org/ontology1#reviewedarticle">
    <skos:related rdf:resource="http://www.example.org/ontology2#article"/>
  </skos:Concept>

</rdf:RDF>
```

49.2 Generating RDF, OWL and WSML from the SEKT Mapping Language API

The Mapping language presented in Section 47.6 is used via a Java API. It provides classes and methods for parsing mapping documents, manipulating the mappings objects in memory and serialising them in various formats. Apart from the original abstract syntax format the API proposes exporting in RDF, OWL-DL and WSML. An example of mapping exported in these various formats is given in this section.

49.2.1 RDF syntax of the mapping language

We first give the RDF triples corresponding to the mapping document header:

```
_"http://sw.deri.org/~francois/mappings/creature2livingThing"
rdf#type
```

```
map#mappingDocument

_"http://sw.deri.org/˜francois/mappings/creature2livingThing"
map#onto1
_"http://sw.deri.org/˜francois/ontologies/o1"

_"http://sw.deri.org/˜francois/mappings/creature2livingThing"
map#onto2
_"http://sw.deri.org/˜francois/ontologies/o2"

_"http://sw.deri.org/˜francois/mappings/creature2livingThing"
dc#creator
_"http://sw.deri.org/˜francois/foaf.rdf"
```

A simple and a more complex mapping rule examples are following. We use for this example *o*1 and *o*2 as shortened namespaces.

```
_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule1"
rdf:type
map#ClassMapping

_"http://sw.deri.org/˜francois/mappings/creature2livingThing"
map#ClassMapping
_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule1"

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule1"
map#directionality
xsd:string^^"bidirectional"

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule1"
map#hasSource
o1#creature

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule1"
map#hasTarget
o2#livingThing

_"http://sw.deri.org/˜francois/mappings/creature2livingThing"
map#ClassMapping
_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule2"

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule2"
map#directionality
xsd:string^^"unidirectional"

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule2"
map#hasSource
o1#Female

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule2"
map#hasTarget
o2#Adult

_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule2"
map#condition
map#attributeValueCondition
```

```
_"http://sw.deri.org/˜francois/mappings/creature2livingThing#rule2"
map#conditionId
xsd:string^^"cond"

xsd:string^^"cond" map#onAttribute o1#age

xsd:string^^"cond" map#conditionOperator map#greaterOrEqual

xsd:string^^"cond" map#conditionValue xsd:integer18
```

49.2.2 Generating OWL

Simple mappings might be expressed in OWL-DL. This gives some restriction since OWL cannot express rules. It is however planned to extend this export using the SWRL extension of OWL.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY base "http://sw.deri.org/˜francois/mappings/Creature2livingThing/">
]>
<rdf:RDF
  xmlns:rdf="&rdf;"
  xmlns:owl="&owl;"
  xmlns:xsd="&xsd;"
  xml:base="&base;">
  <owl:Ontology rdf:about="&base;">
    <owl:imports rdf:resource="http://sw.deri.org/˜francois/ontologies/creature/"/>
    <owl:imports rdf:resource="http://sw.deri.org/˜francois/ontologies/livingthing/"/>
  </owl:Ontology>
  <owl:Class rdf:about="http://ontologies.omwg.org/creature#Creature">
    <rdfs:comment>Map the creature concept to the livingThing concept</rdfs:comment>
    <owl:equivalentClass>
      <owl:Class rdf:about="http://sw.deri.org/˜francois/ontologies/livingthing#Living Thing/">
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://ontologies.omwg.org/livingThing#Adult"/>
    <owl:subClassOf rdf:resource="http://ontologies.omwg.org/creature#Female"/>
  </owl:Class>
</rdf:RDF>
</xml>
```

49.2.3 Generating WSML

WSML (Web Services Modelling Language) is also generated using the export method of a mapping document. It uses a variant of WSML (WSML-DL) which cannot express rules.

```
ontology  _"http://sw.deri.org/˜francois/mappings/Creature2livingThing/"
  nfp
  endnfp
  axiom mappingRuleNumber1
```

```
    nfp
rdfs#label hasValue "Creature to LivingThing"
_"http://purl.org/dc/elements/1.1/description" hasValue "Map the creature concept to
             the livingThing concept"
MappingConfidenceMeasure hasValue "1.0"
    endnfp
  definedBy
?x memberOf _"http://ontologies.omwg.org/livingThing#LivingThing" impliedBy
?x memberOf _"http://ontologies.omwg.org/creature#Creature".

  axiom mappingRuleNumber2
    nfp
     rdfs#label hasValue "Creature to LivingThing"
_"http://purl.org/dc/elements/1.1/description" hasValue "Map the creature concept to
             the livingThing concept"
MappingConfidenceMeasure hasValue "1.0"
    endnfp
  definedBy
    ?x memberOf _"http://ontologies.omwg.org/livingThing#LivingThing" implies
    ?x memberOf _"http://ontologies.omwg.org/creature#Creature".

  axiom mappingRuleNumber3
    nfp
rdfs#label hasValue "conditional female to adult"
MappingConfidenceMeasure hasValue "1.0"
    endnfp
  definedBy
    ?x memberOf _"http://ontologies.omwg.org/livingThing#Adult" impliedBy
    ?x memberOf _"http://ontologies.omwg.org/creature#Female".
```

49.3 Conclusions

Figure 49.1 summarises the transformations presented in this chapter. At first sight, it could seem that the alignment format is the ideal exchange format because it allows to generate, directly or indirectly, so many different formats. However, this is not the case because the SEKT Mapping language is more expressive than the alignment format. The alignment format can only generate simple alignments. So the best option is to enhance the expressiveness of the Alignment format so that it more fully covers the expressive formats. To that extent, the best option is certainly embed the SEKT Mapping language within the Alignment format.

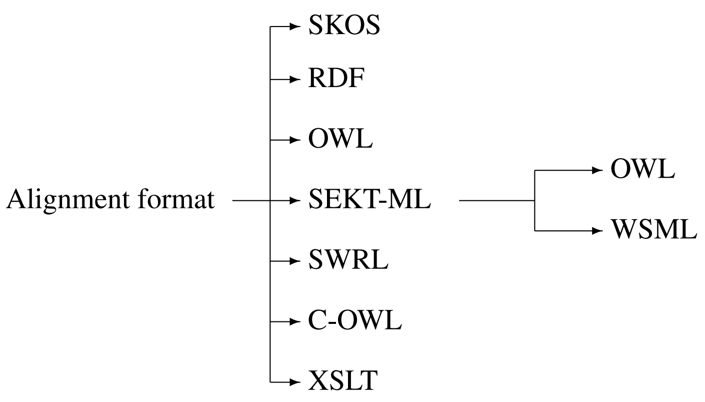


Figure 49.1: Possible export function among alignment formats.

Chapter 50

Abstract syntax and semantics

In this chapter we attempt at going deeper into the format integration satisfying our initial requirements. As mentioned above, the Alignment format made provision for embedding expressive languages in order to suit particular needs. So, instead of confining the users to simple match between ontologies, it can be extended to express more complex alignments. The benefit from the standpoint of applications using this format is that they can still use the simple alignment that may exist.

This chapter describes the language we designed to express ontology alignments. It offers an expressive language following the mapping language of [Scharffe and de Bruijn, 2005] as considered in [Euzenat *et al.*, 2005a] and its semantics is defined independently of any ontology representation language as sketched in [Zimmermann and Euzenat, 2006].

We first introduce an abstract syntax for this language allowing the expression of the examples of the previous chapter. We then explain the principles used for defining a semantics independent from the ontology languages and provide the semantics of this expressive alignment language according to the abstract syntax.

50.1 Abstract syntax

We will define a simple language with maximal expressivity: our goal is to provide the constructors for expressing these correspondences and defining their meaning. Of course, for complexity reasons, it would be natural to reduce the extent of the language. For describing this syntax, we use as much as possible the conventions used for description logics [Baader *et al.*, 2003], however we sometimes divert from them for openness (when a comparator is provided by description logics while in our case it is external to the language) or aesthetic reasons (like avoiding double superscript).

We will consider hereafter that alignments are sets of correspondences satisfying the following grammar. Correspondences (X) themselves are relations between entities (E). They are here restricted to equivalence and subsumption relations as well as membership of an individual (i) to a class (C):

(50.1) $$X := E \equiv E \mid E \sqsubseteq E \mid E \sqsupseteq E$$

(50.2) $$\mid i \in C \mid C \ni i$$

We restricted here the set of relations in order to be more precise in the semantics. However the Alignment format is extensible and new relations can be added, so the definition would rather be:

(50.3) $$X ::= E\ rel\ E$$

For instance enabling to use relations as $fatherOf$ between individuals.

The entities that will be found in correspondences are classes (C), relations (R), properties (P), attributes (A) and instances or individuals (i):

(50.4) $$E ::= C \mid R \mid P \mid A \mid i$$

Separation of ontological entities in five types comes from the ontology mapping language described in [Scharffe and Kiryakov, 2005]. This separation is justified as expression definitions slightly vary depending on this type and their semantics will differ as well. *Class expressions* represent classes or sets of classes linked together via operators. *Property expressions* represent relations whose codomain (or range) is a datatype. *Relations* are standing between two classes. So, the difference between Relation and Properties corresponds to the difference between ObjectProperty and DatatypeProperty in OWL. *Attributes* are relations and properties put in a particular context (see § 1.2.2). Finally, instances of classes can be put in correspondence via *Instance expressions*. To make the distinction between Relations and Properties, we will sometimes use "class relation" and "data property". The structure of expressions varies depending on the expression type. For example, Property expressions may have value restrictions whereas Class expression restrictions are more related to instances or particular attributes of instances. The main construct allows URI of an entity to be directly given to build an expression using constructors. Constructors tell how to group the set of entities given in the expression and are interpreted in model-theoretic terms in § 50.3. Expression constructors can be composed of sub-expressions. Conditions on class, property and relation expressions restrict the scope of the set of entities constructed in the expression.

Alignments relate entities of ontology languages such as OWL, F-logics or others. We consider that these entities (identified as entities in the Alignment format) are typed. The different types that will be considered are the following:

– Classes: c;
– Relations: r;
– Properties: p;
– Instances: i.

Data values and types must also be part of the entities to be considered:

– Data type: d;
– Data value: v;

They are however considered as external to the Alignment language.

From these entities, the Alignment language has constructors for creating more complex expressions. These constructors are the classical boolean algebra expressions (*and*, *or* and *not*) as well as the existence of constraints on class expressions (these constraints being expressed with external operators are not defined like in description logics):

(50.5) $$C ::= c$$
(50.6) $$\mid C \sqcup C \mid C \sqcap C \mid \neg C$$
(50.7) $$\mid \exists K$$

Relation expressions will also be created out of the relations, their boolean combination, constraints on their domain and range, and either their converse relation or their symmetric, transitive or reflexive closures:

(50.8) $R ::= r$

(50.9) $\mid R \sqcup R \mid R \sqcap R \mid \neg R$

(50.10) $\mid \mathrm{dom}(C) \mid \mathrm{range}(C)$

(50.11) $\mid \mathrm{inv}(R)$

(50.12) $\mid \mathrm{sym}(R) \mid \mathrm{trans}(R) \mid \mathrm{refl}(R)$

Property expressions are similar but less complex since they cannot involve properties that only hold for relations (symmetry, transitivity, converse and reflexivity):

(50.13) $P ::= p$

(50.14) $\mid P \sqcup P \mid P \sqcap P \mid \neg P$

(50.15) $\mid \mathrm{dom}(C) \mid \mathrm{range}(d)$

(50.16)

To these are added *attributes*, i.e., properties or relations with a restricted domain. However, these are strictly equivalent to the following expression:

$$A \equiv R \sqcap dom(C) \text{ or } A \equiv P \sqcap dom(C)$$

In order to draw constraints on property and attribute values we introduce the notion of a path which is a sequence of relations possibly empty and possibly ending by a property:

(50.17) $Q ::= Q' \mid p \mid Q'.p$

(50.18) $Q' ::= \epsilon \mid r \mid Q'.r$

Note that we have restricted these paths to atomic steps, i.e., each element of the sequence is identified by a relation or property in the ontology but not a relation expression (R). These paths corresponds to role-value-maps in description logics. They are useful for expressing complex constraints (see below).

The values that can be found verbatim in these constraints can be either data values, individual instances, paths or the do2002a of some operation transf to a set of values:

(50.19) $V ::= v$

(50.20) $\mid i$

(50.21) $\mid Q$

(50.22) $\mid \mathrm{transf}(V*)$

The constraints and operations that can be applied to values are useful to express concrete domain constraints. This was for instance in the definition of § 1.2.3 in which integers are converted into date. The language does consider that these functions (which return a data value) and comparators (which returns a boolean) are external operation with a well-known semantics. As a first set of constraints on datatypes, we consider using datatypes from XQuery (on numeric, string, collections and uri) as well as their comparators. The full set of these comparators is given in Appendix C. This provides:

– 23 functions (fn);
– 14 predicates (cp).

This allows the expression of expressions like:

$$length(collection)\ less\text{-}than\ multiply(integer, integer)$$

Once values are available, the predicates can be evaluated to compare these values. This is the basis of constraints that can be raised against paths for comparing their values, their datatypes or their multiplicity:

(50.23)	$K ::= Q\ cp\ V$	(Value restriction)
(50.24)	$\mid Q\ cp\ d$	(Type restriction)
(50.25)	$\mid \lvert Q \rvert\ cp\ i$	(Multiplicity restriction)

This is used for expressing, e.g., $surname\ subStringOf\ fullname$, $age \in [12\ 16]$ or that $|child| \geq 3$. All these restrictions allow expressing more closely the relation between concepts or properties of different ontologies.

Example 8. *We want to restrict to the wines for which an adjacent region produces a wine made by the same producer. So we define the abstract first path Q as* `locatedIn.adjacentRegion` *and the second one V as* `hasMaker.producesWine.locatedIn`*. The comparison between the values pointed by the first path, and the values pointed by the second one gives the expected result. Abstractly, this is represented like this:*

$$\exists locatedIn.adjacentRegion = hasMaker.producesWine.locatedIn$$

50.2 Ensuring autonomous semantics

In this section, we provide a semantics for the formerly described alignment language. The semantics is based on the satisfaction of correspondences.

Informally, a correspondence $e\ rel_m\ e'$ means: there is a relation, denoted by symbol rel, between the entity described by e and the entity described by e', and our confidence in this relation being valid is equal to m. In fact, to the assertion $e\ rel\ e'$ is assigned a degree of trust m. The formal semantics given here defines the meaning of the former assertion independently of the confidence value. The way this measure is dealt with is left to an external formalism that we do not discuss here.

So, we henceforth consider a correspondence as a triple $\langle e, e', rel\rangle$, composed of:

– two entity expressions e and e';
– a relation symbol rel.

The difficulty in defining the interpretations of entity expressions is due to the fact that these expressions are built upon ontological entities which have there own interpretation in the ontological language [Borgida and Serafini, 2003a]. The semantics of the language of two aligned ontologies can differ widely from each other, and also from the semantics of this alignment language. Therefore, we have proposed a semantics of aligned ontologies that has two levels of interpretation [Zimmermann and Euzenat, 2006], as visualised in Figure 50.1.

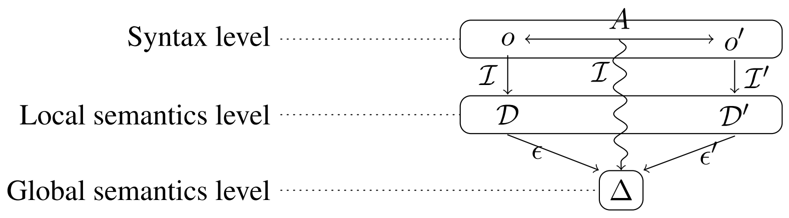


Figure 50.1: Two levels of semantics.

In Figure 50.1, o and o' are two ontologies related with alignment A. $\mathcal{I}$ (resp. $\mathcal{I}'$) denotes an interpretation of o (resp. o') with interpretation domain $\mathcal{D}$ (resp. $\mathcal{D}'$). Since these interpretations and domains may be very different in their structure, we use functions ϵ and ϵ', called equalising functions, to correlate these local domains into a commensurate global domain (Δ), in which alignments are interpreted.

We first give the interpretation of entity expressions independently of the ontological languages semantics (§ 50.3). Then we give the formal semantics of correspondences and define the fundamental notion of a model of aligned ontologies, which encompasses both local and global levels of semantics (§ 50.4).

50.3 Interpreting expressions

We define the full interpretation of entity expressions in the next subsections. The first item is not, properly speaking, part of the semantics. Datatypes, operators, comparators and data transformations are managed by the application, and we will assume that all that is defined in this part can be used in all following definitions. The semantics is given with regard to the abstract syntax.

50.3.1 Datatypes, operators, comparators and transformations

Datatypes are interpreted as in RDF and OWL. Different applications may bring their own datatypes, and their management is quite independent of the semantics. These datatypes are identified in the syntax by:

– Data type: d;
– Data value: v;

A datatype describes the set of values belonging to the type, the set of character strings representing these values and the way to convert these strings to values.

Definition 103 (Datatype)**.** *A* datatype *d is characterised by:*

– *its lexical space $L(d)$ (a set of character strings);*
– *its value space $V(d)$;*
– *a mapping $L2V(d) : L(d) \rightarrow V(d)$ from the lexical space to the value space.*

Definition 104 (Datatype map)**.** *A* datatype map *D is a partial mapping from URI references to datatypes.*

The datatype map allows the retrieval of datatypes from the URI identifying them.

This alignment language uses operators and comparators that are tied to specific datatypes. There must be support for these datatypes when implementing this alignment language.

Interpreting operators: Operators are used to define values that are not explicitly provided by a URI reference or a literal, but that can be calculated with existing values. A URI reference appearing in a `transf` element denotes the functions to be used for this calculation.

Definition 105 (Data operator)**.** *A* data operator *is a triple* $\langle k, (t_i)_{1 \leq i \leq k}, t_r, f\rangle$ *such that:*

- k *is the* arity *of the operator;*
- $\forall 1 \leq i \leq k$*,* t_i *is the* datatype *of the* i^{th} *operand;*
- t_r *is the datatype of the result of the operation;*
- $f : t_1 \times \cdots \times t_k \rightarrow t_r$ *is a function.*

Definition 106 (Operator map)**.** *An* operator map *is a partial map from URI references to data operators.*

Interpreting comparators: Comparators serve to compare two values or elements. In fact, a comparator is a particular case of operator, namely a binary operator with a resulting domain being boolean.

Definition 107 (Comparator)**.** *A* comparator *is a data operator* $cp = \langle 2, (t, t'), \text{Bool}, f\rangle$.

Operator map is also generalised from comparator map. Table C.2 in Appendix gives a library of common comparators.

Interpreting data transformation: Data transformations are more complicated functions that are used to convert data into other data, in order to render possible the mediation between two conceptually similar system with divergent concrete representations. A data transformation appears in the semantics just as an operator, where the operands are composed of the input data and the parameters of the function, and the result of the operation is the output of the function.

The implementation of a data transformation can exist outside the application, for example as a web service. So, in the semantics, it is interpreted as a function.

In the remainder, it will be assumed that datatypes, operators and comparators are provided by the application, and a datatype map D, a comparator map $g. : \text{cp} \mapsto g_{\text{cp}}$ and a transformation/operator map $h. : \text{transf} \mapsto h_{\text{transf}}$ are supposed to exist when we define the semantics.

Similarly to datatypes, different applications can implement different operators so that the list of interpretable operators can be extended. As an example, we provide a library of data operators in Table C.1 in Appendix C.

50.3.2 Interpreting literals

Literal values v are interpreted according to a datatype map D.

Definition 108 (Literals interpretation)**.** *Let* D *be a datatype map. Let* $\mathcal{L}$ *be a set of literals. An interpretation of* $\mathcal{L}$ *is a pair* $\mathcal{I}_L = \langle \mathbb{D}, \cdot^{\mathcal{I}_L}\rangle$ *with* $\mathbb{D}$ *the domain of interpretation of literals and* $\cdot^{\mathcal{I}_L} : \mathcal{L} \rightarrow \mathbb{D}$ *such that:*

– $(\texttt{"}v\texttt{"})^{\mathcal{I}_L} = \texttt{"}v\texttt{"}$.[1], *i.e., plain literals are interpreted as themselves,*
– $(\texttt{"}v\texttt{"\^{}\^{}}d)^{\mathcal{I}_L} = L2V(D(d))(v)$ *if* $v \in L(D(d))$

If a literal is ill-formed (like, for instance, `true^^&xsd;integer`) then no interpretation can exist.

50.3.3 Interpreting URI references

The basic notion of interpretation of URI references is very liberal. Indeed, given a set $\mathcal{U}$ of URI references, an interpretation of $\mathcal{U}$ is just a mapping from the elements of $\mathcal{U}$ to a domain of interpretation. More formally:

Definition 109 (URIref interpretation)**.** *A* URIref interpretation *of a set $\mathcal{U}$ of URI references is a pair $\mathcal{I}_U = \langle \Delta, \cdot^{\mathcal{I}_U} \rangle$ such that Δ is a set called the global domain of interpretation and $\cdot^{\mathcal{I}_U} : \mathcal{U} \rightarrow \Delta$.*

Note that the set Δ (the global domain of interpretation) may contain elements that are themselves sets of elements, or sets of pairs or even sets of sets.

50.3.4 Interpreting path expressions

A path is essentially the same as a relation, so it must be interpreted as a set of pairs of elements.

It is defined by a (potentially empty) sequence of relations possibly ended by a property:

$$Q ::= Q' \mid p \mid Q'.p$$
$$Q' ::= \epsilon \mid r \mid Q'.r$$

In the abstract syntax, ϵ denotes the empty path, r denotes a relation and p a property represented as a URI reference.

Definition 110 (Path interpretation)**.** *A* path interpretation *is a triple $\mathcal{I}_P = \langle \mathcal{D}, \mathcal{I}_L, \cdot^{\mathcal{I}_P} \rangle$ with $\mathcal{D}$ a set called the* object domain *and $\mathcal{I}_L$ a literal interpretation such that:*

– $\epsilon^{\mathcal{I}_P} = \mathcal{D} \times \mathcal{D}$;
– $r^{\mathcal{I}_P} \subseteq \mathcal{D} \times \mathcal{D}$;
– $(Q'.r)^{\mathcal{I}_P} = \{\langle x, z\rangle \in \mathcal{D} \times \mathcal{D} / \exists y \in \mathcal{D}, \langle x, y\rangle \in Q'^{\mathcal{I}_P} \wedge \langle y, z\rangle \in r^{\mathcal{I}_P}\}$;
– $p^{\mathcal{I}_P} \subseteq \mathcal{D} \times \mathbb{D}$;
– $(Q.p)^{\mathcal{I}_P} = \{\langle x, z\rangle \in \mathcal{D} \times \mathbb{D} / \exists y \in \mathcal{D}, \langle x, y\rangle \in Q^{\mathcal{I}_P} \wedge \langle y, z\rangle \in p^{\mathcal{I}_P}\}$.

As we will see, this definition introduces a necessary constraint on the URIref interpretation: URI references representing a relation should be interpreted as a set of pairs.

[1]We consistently use the same notations for each kind of interpretation, so that we do not redefine existentially all their components in further definition. For instance, when a definition defines a literal interpretation $\mathcal{I}_L$, it will be assumed that the set $\mathbb{D}$ and function $\cdot^{\mathcal{I}_L}$ are also defined.

50.3.5 Interpreting paths or values

In the abstract syntax, v is a literal interpreted as a value, i is a URI reference interpreted as an individual, Q is a path expression and $\mathtt{transf}$ denotes an operator, given by a URI reference. It is assumed that the number of operands in the parenthesis is equal to the arity of the associated operator. Since a path or value can take several different forms (simple value, URI reference, path or operation), it embeds the necessary interpretations for all these variants, according to the definitions given above.

$$\begin{aligned} V ::= &\ v \\ &\mid i \\ &\mid Q \\ &\mid \mathrm{transf}(V*) \end{aligned}$$

The interpretation of paths and values is more difficult because we authorise paths in the expression of values. Paths are interpreted as relations. We need to keep these relations intact, for instance for comparing the value at the end of two paths from a single instance. So, even though it may seem unnatural, we have chosen to interpret all values as a set of pairs.

Definition 111 (Path or value interpretation)**.** *A* path or value interpretation *is a tuple* $\mathcal{I}_V = \langle \mathcal{I}_U, \mathcal{I}_P, \cdot^{\mathcal{I}_V} \rangle$ *with* $\mathcal{I}_P$ *a path interpretation built upon a literal interpretation* $\mathcal{I}_L$ *such that:*

- $v^{\mathcal{I}_V} = \mathcal{D} \times \{v^{\mathcal{I}_L}\}$;
- $i^{\mathcal{I}_V} = \mathcal{D} \times \{i^{\mathcal{I}_U}\}$;
- $Q^{\mathcal{I}_V} = Q^{\mathcal{I}_P}$;
- $\mathrm{transf}(V_1, \dots, V_n)^{\mathcal{I}_V} = \{\langle x, h_{\mathrm{transf}}(y_1, \dots, y_n)\rangle / \langle x, y_1\rangle \in V_1^{\mathcal{I}_V} \wedge \dots \wedge \langle x, y_n\rangle \in V_n^{\mathcal{I}_V}\}$.

By consistently interpreting paths and values as pairs of elements, we do not need to separate cases in the definitions below.

50.3.6 Interpreting restrictions

Restrictions are of three types: value restrictions, type restrictions and multiplicity restrictions. Whatever the restriction, it corresponds to a class definition. Each type of restriction is interpreted differently from the others, but they share the same abstract syntax:

$$\begin{aligned} K ::= &\ Q\ cp\ V && \text{(Value restriction)} \\ &\mid Q\ cp\ d && \text{(Datatype restriction)} \\ &\mid Q\ cp\ n && \text{(Multiplicity restriction)} \end{aligned}$$

The comparator is denoted by cp and is identified by a URI reference. Q is a path expression and V is a path or value expression.

Interpreting value restrictions: A value restriction gives the class of individuals for which the comparison *cp* holds between a value at the end of the path Q and a value denoted by V.

Definition 112 (Value restriction interpretation)**.** *A* value restriction interpretation *is a tuple $\mathcal{I}_{VR} = \langle \mathcal{I}_V, \cdot^{\mathcal{I}_{VR}} \rangle$ with $\mathcal{I}_V$ a path or value interpretation with an object domain $\mathcal{D}$ such that:*

– $(Q \text{ cp } V)^{\mathcal{I}_{VR}} = \{x \in \mathcal{D} / \exists y, y' \in \mathcal{D} \cup \mathbb{D}, \langle x, y \rangle \in Q^{\mathcal{I}_V} \wedge \langle x, y' \rangle \in V^{\mathcal{I}_V} \wedge g_{\text{cp}}(y, y')\}.$

with g_{cp} the data operator associated with cp

Interpreting type restrictions: In this case, d denotes one or several datatypes. The type restriction imposes that the values pointed by the path Q belongs to the datatypes identified by d.

Definition 113 (Type restriction interpretation)**.** *A* type restriction interpretation *is a tuple $\mathcal{I}_{TR} = \langle \mathcal{I}_V, \cdot^{\mathcal{I}_{TR}} \rangle$ with $\mathcal{I}_V$ a path or value interpretation with an ovbject domain $\mathcal{D}$ such that:*

– $(Q \text{ cp } d)^{\mathcal{I}_{TR}} = \{x \in \mathcal{D} / \forall y \in \mathcal{D} \cup \mathbb{D}, \langle x, y \rangle \in Q^{\mathcal{I}_V} \Rightarrow g_{\text{cp}}(y, V(D(d)))\}.$

with g_{cp} the data operator associated with cp

Interpreting occurrence restrictions: In this case, V denotes an integer (or a set of integers), which is meant to denote a cardinality. The comparator compares this number to the cardinality of the attribute denoted by the path Q.

Definition 114 (Occurrence restriction interpretation)**.** *An* occurrence restriction interpretation *is a tuple $\mathcal{I}_{OR} = \langle \mathcal{I}_V, \cdot^{\mathcal{I}_{OR}} \rangle$ with $\mathcal{I}_V$ a path or value interpretation with an object domain $\mathcal{D}$ such that:*

– $(Q \text{ cp } n)^{\mathcal{I}_{OR}} = \{x \in \mathcal{D} / g_{\text{cp}}(|\{y \in \mathcal{D} \cup \mathbb{D}; \langle x, y \rangle \in Q^{\mathcal{I}_V}\}|, n^{\mathcal{I}_L})\}.$

with g_{cp} the data operator associated with cp

Comparators for occurrence restrictions are maxCardinality ($\leq$), minCardinality ($\geq$) or cardinality ($=$). We define a *restriction interpretation* as follows:

Definition 115 (Restriction interpretation)**.** *A restriction interpretation $\mathcal{I}_K$ is the combination of a value restriction interpretation, a type restriction interpretation and an occurrence restriction interpretation having the same path or value interpretation.*

50.3.7 Interpreting class expressions

Classes are interpreted as sets of elements. Class constructors have a very common semantics which is exactly the basic description logic interpretation.

$$\begin{aligned} C ::= & \; c \\ & \mid C \sqcup C \mid C \sqcap C \mid \neg C \\ & \mid \exists K \end{aligned}$$

The single c denotes a class represented by a URI reference. C denotes a class expression. The constructors are represented in a typical description logics fashion ($\sqcup$ for union/or, $\sqcap$ for intersection/and, $\neg$ for negation/not).

Definition 116 (Class interpretation)**.** *A* class interpretation *is a tuple $\mathcal{I}_C = \langle \mathcal{I}_K, \cdot^{\mathcal{I}_C} \rangle$, with $\mathcal{I}_K$ a restriction interpretation with object domain $\mathcal{D}$, such that:*

- $c^{\mathcal{I}_C} \subseteq \mathcal{D}$;
- $(C_1 \sqcup C_2)^{\mathcal{I}_C} = C_1^{\mathcal{I}_C} \cup C_2^{\mathcal{I}_C}$;
- $(C_1 \sqcap C_2)^{\mathcal{I}_C} = C_1^{\mathcal{I}_C} \cap C_2^{\mathcal{I}_C}$;
- $(\neg C)^{\mathcal{I}_C} = \mathcal{D} \setminus C^{\mathcal{I}_C}$;
- $(\exists K)^{\mathcal{I}_C} = K^{\mathcal{I}_K}$.

50.3.8 Interpreting relation expressions

Class properties are interpreted as set-theoretic relations, i.e., sets of pairs of elements.

$$\begin{aligned} R ::= &\ r \\ &\mid R \sqcup R \mid R \sqcap R \mid \neg R \\ &\mid \mathrm{dom}(C) \mid \mathrm{range}(C) \\ &\mid \mathrm{inv}(R) \\ &\mid \mathrm{sym}(R) \mid \mathrm{trans}(R) \mid \mathrm{refl}(R) \end{aligned}$$

In this abstract syntax, r is a single URI reference, that denotes a relation between instances. Symbol R denotes a (class) relation. Constructors are given in a Description-Logic-like syntax: union ($\sqcup$), intersection ($\sqcap$), complement ($\neg$), range restriction (range), domain restriction (dom). C represents a class. Finally, inv, sym, trans and refl denotes the inverse, the symmetric closure, the transitive closure and reflexive closure respectively.

Definition 117 (Relation interpretation)**.** *A* Relation interpretation *is a tuple $\mathcal{I}_{CP} = \langle \mathcal{I}_C, \cdot^{\mathcal{I}_{CP}} \rangle$, with $\mathcal{I}_C$ a class interpretation, built on path interpretation $\mathcal{I}_P$ with object domain $\mathcal{D}$, such that:*

- $r^{\mathcal{I}_{CP}} = r^{\mathcal{I}_P}$;
- $(R_1 \sqcup R_2)^{\mathcal{I}_{CP}} = R_1^{\mathcal{I}_{CP}} \cup R_2^{\mathcal{I}_{CP}}$;
- $(R_1 \sqcap R_2)^{\mathcal{I}_{CP}} = R_1^{\mathcal{I}_{CP}} \cap R_2^{\mathcal{I}_{CP}}$;
- $(\neg R)^{\mathcal{I}_{CP}} = \mathcal{D} \times \mathcal{D} \setminus R^{\mathcal{I}_{CP}}$;
- $\mathrm{range}(C)^{\mathcal{I}_{CP}} = \mathcal{D} \times C^{\mathcal{I}_C}$;
- $\mathrm{dom}(C)^{\mathcal{I}_{CP}} = C^{\mathcal{I}_C} \times \mathcal{D}$;
- $\mathrm{inv}(R)^{\mathcal{I}_{CP}} = \{\langle x, y \rangle / \langle y, x \rangle \in R^{\mathcal{I}_{CP}}\}$;
- $\mathrm{sym}(R)^{\mathcal{I}_{CP}} = R^{\mathcal{I}_{CP}} \cup \mathrm{Inv}(R)^{\mathcal{I}_{CP}}$;
- $\mathrm{refl}(R)^{\mathcal{I}_{CP}} = R^{\mathcal{I}_{CP}} \cup \{\langle x, x \rangle / x \in \mathcal{D}\}$;
- $\mathrm{trans}(R)^{\mathcal{I}_{CP}} = R^{\mathcal{I}_{CP}} \cup \{\langle x, z \rangle / \exists y \in \mathcal{D}, \langle x, y \rangle \in R^{\mathcal{I}_{CP}} \wedge \langle y, z \rangle \in \mathrm{trans}(R)^{\mathcal{I}_{CP}}\}$;

50.3.9 Interpreting data property expressions

Data properties are also interpreted as set-theoretic relations, but since the codomain has to be a datatype, values are restricted to datatype values.

$$\begin{aligned}P ::= &\ p\\ &\mid P \sqcup P \mid P \sqcap P \mid \neg P\\ &\mid \mathrm{dom}(C) \mid \mathrm{range}(d)\end{aligned}$$

This syntax and semantics are similar to those of relations.

Definition 118 (Data property interpretation)**.** *A* data property interpretation *is a tuple $\mathcal{I}_{DP} = \langle \mathcal{I}_C, \cdot^{\mathcal{I}_{DP}} \rangle$, with $\mathcal{I}_C$ a class interpretation, built on path or value interpretation $\mathcal{I}_V$ with object domain $\mathcal{D}$, such that:*

- $p^{\mathcal{I}_{DP}} = p^{\mathcal{I}_V}$;
- $(P_1 \sqcup P_2)^{\mathcal{I}_{DP}} = P_1^{\mathcal{I}_{DP}} \cup P_2^{\mathcal{I}_{DP}}$;
- $(P_1 \sqcap P_2)^{\mathcal{I}_{DP}} = P_1^{\mathcal{I}_{DP}} \cap P_2^{\mathcal{I}_{DP}}$;
- $(\neg P)^{\mathcal{I}_{DP}} = \mathcal{D} \times \mathbb{D} \setminus P^{\mathcal{I}_{DP}}$;
- $(\mathrm{dom}(C))^{\mathcal{I}_{DP}} = C^{\mathcal{I}_C} \times \mathcal{D}$;
- $(\mathrm{range}(d))^{\mathcal{I}_{DP}} = \{\langle x, y\rangle \in P^{\mathcal{I}_{DP}} / y \in L(D(d))\}$;

Attribute definitions that have been introduced in §46.3.2 can be expressed in first order logic as:

$$C(x) \wedge P(x, y)$$

which corresponds to:

$$dom(C) \sqcap P$$

50.3.10 Interpreting instance expressions

Instances are always identified by a URI reference. They are interpreted by a URIref interpretation (see §50.3.3).

50.3.11 Interpreting entity expressions

This is the general interpretation of all expressions of the language. An entity expression is of one of the five following types (instance, class, relation, property and attribute):

$$E ::= C \mid R \mid P \mid A \mid i$$

The abstract syntax shall be understood as: C for class expression, R for class relation expression, P for data property expression, A for attribute expression and i for instance expression.

Definition 119 (Expression interpretation)**.** *An* expression interpretation *is a tuple*

$$\mathcal{I} = \langle \Delta, \mathcal{D}, \mathbb{D}, \mathcal{I}_L, \mathcal{I}_U, \mathcal{I}_P, \mathcal{I}_V, \mathcal{I}_K, \mathcal{I}_C, \mathcal{I}_{CP}, \mathcal{I}_{DP}, \cdot^{\mathcal{I}} \rangle$$

with:

– $\mathcal{I}_L = \langle \mathbb{D}, \cdot^{\mathcal{I}_L} \rangle$ *a literal interpretation;*
– $\mathcal{I}_U = \langle \Delta, \cdot^{\mathcal{I}_U} \rangle$ *a URIRef interpretation;*
– $\mathcal{I}_P = \langle \mathcal{D}, \mathcal{I}_L, \cdot^{\mathcal{I}_P} \rangle$ *a path interpretation;*
– $\mathcal{I}_V = \langle \mathcal{I}_U, \mathcal{I}_P, \cdot^{\mathcal{I}_V} \rangle$ *a path or value interpretation;*
– $\mathcal{I}_K = \langle \mathcal{I}_V, \cdot^{\mathcal{I}_K} \rangle$ *a restriction interpretation;*
– $\mathcal{I}_C = \langle \mathcal{I}_K, \cdot^{\mathcal{I}_C} \rangle$ *a class interpretation;*
– $\mathcal{I}_{CP} = \langle \mathcal{I}_C, \cdot^{\mathcal{I}_{CP}} \rangle$ *a relation interpretation;*
– $\mathcal{I}_{DP} = \langle \mathcal{I}_C, \cdot^{\mathcal{I}_{DP}} \rangle$ *a data property interpretation;*

such that:

– $E^{\mathcal{I}} \in \Delta$;
– $i^{\mathcal{I}} = i^{\mathcal{I}_U}$;
– $c^{\mathcal{I}} = c^{\mathcal{I}_U}$;
– $p^{\mathcal{I}} = p^{\mathcal{I}_U}$;
– $r^{\mathcal{I}} = r^{\mathcal{I}_U}$.
– $C^{\mathcal{I}} = C^{\mathcal{I}_C}$;
– $R^{\mathcal{I}} = R^{\mathcal{I}_{CP}}$;
– $P^{\mathcal{I}} = P^{\mathcal{I}_{DP}}$.

The two entities appearing in a correspondence are interpreted with this definition. Once these two interpretations are known, they have to be compared according to the relation given in the correspondence. This comparison will determine the validity of the interpretation with regard to the correspondence.

50.4 Interpreting correspondences

In order to relate the semantics of aligned ontologies to the semantics of our defined language, we first give general notions of model-theoretic semantics that should encompass most of the language semantics that are used to represent ontologies. So, we first have to define an interpretation of an ontology, in model-theoretic terms.

Definition 120 (Interpretation of an ontology)**.** *Given an ontology o, an interpretation m of o is a pair $\langle \mathcal{I}, \mathcal{D} \rangle$ such that $\mathcal{D}$ is a set called the domain of interpretation and $\mathcal{I}$ is a function from elements of o to elements of a domain of interpretation $\mathcal{D}$.*

Among interpretations, there are particular ones that are said to *satisfy* the ontology. The local semantics of ontologies determine the satisfaction relation $\models$ that relates interpretations to satisfied ontologies, *i.e.*, $m \models o$ if and only if m satisfies o. The interpretations that satisfy o are called the models of o and denoted by $\mathrm{Mod}(o)$.

In this language, ontological entities appear in the form of URI references denoting instances, classes or properties. These entities have an interpretation in both the ontology language and the alignment language semantics. In order to connect the alignment interpretation to the local (ontological) interpretations, the simplest solution would be to define the expression interpretation of these entities as being equal to their local interpretation. Unfortunately, the diversity of formalism and conceptualisation is such that it is often impossible to interpret two ontologies in the same global domain.

Therefore, in order to assess the validity of a relation between heterogeneous ontologies, we have introduced the notion of *equalising function* [Zimmermann and Euzenat, 2006], which serves to make the domains commensurate.

Definition 121 (Equalising function)**.** *Given a family of interpretations $(m_i)_{i \in I} = \langle \mathcal{D}_i, \mathcal{I}_i \rangle_{i \in I}$ of local ontologies, an* equalising function *for $(m_i)_{i \in I}$ is a family of functions $(\epsilon_i)_{i \in I}$ from the local domains of interpretation $\mathcal{D}_i$ to a global domain Δ (i.e., for all $i \in I$, $\epsilon_i : \mathcal{D}_i \to \Delta$).*

So equalising functions not only define a global domain for the interpretation of a set of ontologies, but also define how local domains are correlated in the global interpretation.

Example 9. *Many semantics impose instances to be interpreted as atomic elements of a domain, while classes are sets of atomic elements. In such cases, the equivalence of an instance and a class would be unsatisfiable without equalising function.*

The ultimate goal of the semantics is to define the satisfaction of aligned ontologies. We give the following preliminary definition.

Definition 122 (Aligned ontologies)**.** *The structure made of two ontologies o and o' and an alignment A between these ontologies is called aligned ontologies and denoted by $A(o, o')$.*

In order to interpret aligned ontologies, we not only need a global interpretation of the ontologies, but also an interpretation function that complies with the constructors and operators of the mapping language.

Definition 123 (Interpretation of aligned ontologies)**.** *Let o and o' be two ontologies aligned with alignment A. An interpretation of $A(o, o')$ is a triple $\langle \mathcal{I}, \mathcal{M}, \epsilon \rangle$ composed of:*

- *an expression interpretation $\mathcal{I}$ having URI interpretation $\mathcal{I}_U$;*
- *a pair $\mathcal{M} = \langle m, m' \rangle$ of local models where $m \in \mathrm{Mod}(o)$ and $m' \in \mathrm{Mod}(o')$;*
- *an equalising function $\epsilon = \langle \epsilon, \epsilon' \rangle$ for $\mathcal{M}$ to a global domain Δ.*

Moreover, for all URI reference u appearing in the first (respectively second) entity expressions of A, $u^{\mathcal{I}_U} = \epsilon(u^m)$ (respectively $u^{\mathcal{I}_U} = \epsilon'(u^{m'})$.)

As noted in § 50.2, a correspondence is denoted by a triple $\langle E_1, E_2, rel \rangle$. Its syntax is:

$$\begin{aligned} X {:=}\; & E \equiv E \mid E \sqsubseteq E \mid E \sqsupseteq E \\ & \mid i \in C \mid C \ni i \end{aligned}$$

or more generally:

$$X ::= E \; rel \; E$$

Interpretation of correspondences depends on the interpretation of relations. The interpretation of the symbol that denotes a correspondence relation does not vary from one alignment interpretation to the other. A relation symbol rel is interpreted as a binary relation $\widetilde{rel}$.

The satisfaction of a correspondence is an important notion of the semantics. Indeed, a correspondence in an alignment acts as an axiom in an ontology: it allows discriminating valid and invalid interpretations. Consequently, it serves to define a model of aligned ontologies.

Definition 124 (Satisfied correspondence)**.** *Let $c = \langle e, e', rel\rangle$ be a correspondence of an alignment A between ontologies o and o'. c is satisfied by an interpretation $\langle \mathcal{I}, \mathcal{M}, \epsilon\rangle$ of $A(o, o')$ if and only if $\langle e^{\mathcal{I}}, e'^{\mathcal{I}}\rangle \in \widetilde{rel}$. This is written $\mathcal{I} \models c$.*

For instance, for the relations used in the above X grammar, the interpretation can be the following:

$$\mathcal{I} \models e \equiv e' \text{ if and only if } e^{\mathcal{I}} = e'I$$
$$\mathcal{I} \models e \sqsubseteq e' \text{ if and only if } e^{\mathcal{I}} \subseteq e'I$$
$$\mathcal{I} \models e \sqsupseteq e' \text{ if and only if } e^{\mathcal{I}} \supseteq e'I$$
$$\mathcal{I} \models i \in C \text{ if and only if } i^{\mathcal{I}} \in C^{\mathcal{I}}$$
$$\mathcal{I} \models C \ni i \text{ if and only if } i^{\mathcal{I}} \in C^{\mathcal{I}}$$

In which $\tilde{\equiv}$ is $=$, $\tilde{\sqsubseteq}$ is $\subseteq$, $\tilde{\sqsupseteq}$ is $\supseteq$, $\tilde{\in}$ is $\in$, and $\tilde{\ni}$ is $\ni$,

If all correspondences of an alignment are satisfied, then the interpretation is called a model of the alignment.

Definition 125 (Model of aligned ontologies)**.** *A* model of aligned ontologies $A(o, o')$ *is an interpretation of $A(o, o')$ that satisfies all the correspondences of A. If a model is written $\mathcal{M}$, then it is noted $\mathcal{M} \models A$.*

From these notions, it is possible to define the set of consequences of aligned ontologies [Zimmermann and Euzenat, 2006].

50.5 Synthesis

We have provided an abstract syntax and semantics for an expressive alignment language. They are summarised in Appendix B in a single table.

The specificity of this semantics is its capacity to interpret systems relating heterogeneous formalisms. So it is a technical mean to offer an expressive language that can bridge between heterogeneous ontology languages and yet take this semantics into account.

This semantics covers the rather rich set of constructors and operators that gives it an elaborate expressiveness. Examples gave a hint of the possibilities offered by the language. However, this expressiveness is achieved at the cost of a hard reasoning procedure. Indeed, since an instance can be interpreted as a class or a relation, and class complement and relation complement exist, this language is most likely to have undecidable satisfiability.

Chapter 51

Operational syntaxes

The abstract syntax is sufficient for defining the semantics of the language. It role is to support inductive semantic definitions. However it is not sufficient for using in operational applications. First, it is ambiguous: when a URI is used in this syntax, it is not always possible to decide if it is a property or a relation, a class or an individual. This propagates through a number of operators. So, it is preferable to design a syntax which resolves this ambiguity. Then, the abstract syntax only focuses on the expression of correspondences and does not cover many other useful information about alignments such as the identification of the alignment and the matched ontologies or metadata such as the method used for establishing the alignment.

Hence, in order to be used by applications, we provide below two concrete syntax:

The exchange syntax (§51.1) is used for exchange of alignments between different systems. It is an extension of the Alignment format [Euzenat, 2004]. This syntax is expressed in RDF/XML so that it can be parsed easily and have standard serialisation methods for being interchanged through networks.

The surface syntax (§51.2) is meant to be more easily read by human users and extends the "human readable" syntax of [Scharffe and de Bruijn, 2005]. It addresses the problem of displaying, and expressing, alignments in a less verbose format than RDF/XML.

Both syntax are designed to be equivalent and to correspond to the abstract syntax presented in Chapter 50.

51.1 Exchange syntax

In order to be exchanged in tools, this language requires a concrete syntax. We provide below a first concrete syntax dedicated to the exchange of information between systems. As is most practical in modern computing this syntax is an XML syntax. However, in order to be as compliant as possible with semantic web technology, this XML syntax follows the RDF/XML rules and an ontology describing the concepts used in this RDF/XML syntax has been defined (and is available in Appendix D).

This syntax combines the alignment format of [Euzenat, 2004] and the OMWG mapping language of [Scharffe and de Bruijn, 2005] as considered in [Euzenat *et al.*, 2005a]. It takes from the first the XML syntax and structure, and embeds the complex expression constructs expressible in the second.

51.1.1 General structure

This syntax has been defined from the Alignment format [Euzenat, 2004], the OMWG Ontology mapping language[1] and our own deliverable 2.2.6 [Euzenat *et al.*, 2005b]. The language follows (and extends) the Alignment format structure until the entity level in which the specific entity constructs provided in this deliverable have been defined.

The syntax is here presented under a Backus-Naur form, though it is in reality order independent, i.e., the order of appearance of attributes or elements does not matter.

The default namespace applying to elements and attributes in the following grammar descriptions is `omwg` standing for `http://www.owmg.org/TR/d7/d7.2/`, `align` is equivalent to `http://knowledgeweb.semanticweb.org/heterogeneity/alignment/`.

The structure of the alignment is the same as that of the Alignment format:

```
⟨alignment⟩ ::= <align:Alignment rdf:about="⟨uri⟩">
                 ⟨annotation⟩*
                 <align:level>2OMWG</align:level>
                 <align:onto1>⟨onto⟩</align:onto1>
                 <align:onto2>⟨onto⟩</align:onto2>
                 (<align:map>⟨cell⟩</align:map>)*
                </align:Alignment>
```

This structure contains information which is not featured in the abstract syntax (because it has no impact on the definiton of the semantics). Among the annotations used by this format, a very important one is the definition of the alignment level. This is a string which in the case of the expressive language should be ”2OMWG”. This tells tools that the alignment is on level 2, i.e., correspondences are across constructed entities, and that the corresponding entities are specified according to this document. This ensures the compatibility with other extensions of the format.

As this format allows defining correspondences between ontologies written in different languages the declaration of the two aligned ontologies includes the identification of the formalism used to represent them. This has been added to the Alignment format. The `Ontology` element gather into one place the information about the ontologies:

```
⟨onto⟩       ::= <align:Ontology rdf:about="⟨uri⟩">
                  <align:location> ⟨url⟩ </align:location>
                  <align:formalism> ⟨formalism⟩ </align:formalism>
                  ⟨description⟩
                 </align:Ontology>

⟨formalism⟩ ::= <align:Formalism align:uri="⟨uri⟩" align:name="⟨string⟩" />
```

The alignment itself is structured as a set of cells, each representing a correspondence between two entity expressions:

```
⟨cell⟩       ::= <align:Cell rdf:about="⟨uri⟩">
                  ⟨annotation⟩*
                  <align:entity1>⟨entity⟩</align:entity1>
```

[1] http://www.omwg.org/TR/d7/

```
      <align:entity2>⟨entity⟩</align:entity2>
      <align:measure>⟨value⟩</align:measure>
      <align:relation>⟨relation⟩</align:relation>
    </align:Cell>
```

When the alignment results from an algorithm, each correspondence may be discovered with a certain degree of certainty. It might be the case that the algorithm uses a set of rules giving an evidence that two entities must be related to a certain extend. The ⟨*measure*⟩ reflects the confidence given to each correspondence. It takes values between 0 and 1.

The type of entities being part of the correspondence and the relation standing between them (equivalence or subsumption) are represented by the ⟨*relation*⟩.

The list of relations defined in our language is currently the following:

```
⟨relation⟩ ::= Equivalence | Subsumes | SubsumedBy
             | InstanceOf | HasInstance.
```

Here is an example of an alignment using only simple entities as in the previous Alignment format:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE rdf:RDF SYSTEM "align.dtd" [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY wine "http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#">
  <!ENTITY vin "http://ontology.deri.org/vin#">
]>

<rdf:RDF xmlns="http://knowledgeweb.semanticweb.org/heterogeneity/alignment"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:owmg="http://www.owmg.org/TR/d7/d7.2/"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
 <Alignment>
    <xml>yes</xml>
    <dc:creator rdf:resource="http://sw.deri.org/˜francois/"/>
    <dc:date>2006/06/07</dc:date>
    <method>manual</method>
    <purpose>example</purpose>
    <level>0</level>
    <type>**</type>
    <onto1>
      <Ontology rdf:about="&wine;">
        <location>http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine</location>
        <formalism>
          <Formalism name="OWL 1.0" uri="http://www.w3.org/2002/07/owl#"/>
        </formalism>
      </Ontology>
    </onto1>
    <onto2>
      <Ontology rdf:about="&vin;">
        <location>http://sw.deri.org/˜francois/ontologies/OntologyDuVin.wsml</location>
        <formalism>
          <Formalism name="WSML"
                     uri="http://www.wsmo.org/wsml/wsml-syntax/wsml-dl"/>
        </formalism>
      </Ontology>
    </onto2>
    <map>
      <Cell>
        <entity1 rdf:resource="&wine;VintageYear"/>
        <entity2 rdf:resource="&vin;Millesime"/>
        <measure rdf:datatype="&xsd;float">1.0</measure>
        <relation>equivalence</relation>
      </Cell>
      <Cell>
        <entity1 rdf:resource="&wine;yearValue"/>
        <entity2 rdf:resource="&vin;annee_millesime"/>
        <measure rdf:datatype="&xsd;float">1.0</measure>
        <relation>equivalence</relation>
      </Cell>
      <Cell>
        <entity1 rdf:resource="&wine;locatedIn"/>
        <entity2 rdf:resource="&vin;hasTerroir"/>
        <measure rdf:datatype="&xsd;float">1.0</measure>
        <relation>subsumed</relation>
      </Cell>
      <Cell>
        <entity1 rdf:resource="&wine;White"/>
        <entity2 rdf:resource="&vin;Blanc"/>
        <measure rdf:datatype="&xsd;float">1.0</measure>
        <relation>equivalence</relation>
      </Cell>
    </map>
  </Alignment>
```

Simple expressions are used in the following example. More complex ones will be studied later in this section. In `Vin`, `Blanc` (French word for white) is a class that contains the variety of white-yellow colours like golden, pale yellow, and so forth.

This example, is only a level 0 example because it does not consider complex entities as defined by our level 2 language.

A benefit of this language is its ability of modelling complex correspondences. Expressions nested in cells express such correspondences and will be further described.

There are basically five types of entities: Classes, Properties, Relations, Attributes and Instances.

$$\begin{array}{lcl}
\langle \textit{entity} \rangle & ::= & \langle \textit{classexpr} \rangle \\
 & | & \langle \textit{relexpr} \rangle \\
 & | & \langle \textit{propexpr} \rangle \\
 & | & \langle \textit{attexpr} \rangle \\
 & | & \langle \textit{instance} \rangle
\end{array}$$

They can be simply identified by their URIs like in the genuine Alignment format or be composed within the format: this is the additional expressiveness that this format provides. We use boolean constructors (`And`, `Or` and `Not`) and relation constructors (`Inverse`, `Transitive`, `Symmetric` and `Reflexive` closures). Relations and properties allows one to distinguish whether an attribute has class instances in its codomain (Relations) or datatype values (Properties).

This distinction is needed because several relation constructors are not applicable to (datatype) properties.

We describe the different expressions in the remaining subsections.

51.1.2 Class expression

A class expression allows the specification of complex classes by grouping them using sets operators and by applying restrictions to those sets. We distinguish between three operators for class expressions: ⟨*not*⟩, ⟨*and*⟩, ⟨*or*⟩. Their semantics is comparable to the semantic of difference, union, intersection in set theory.

```
⟨classexpr⟩ ::= <Class rdf:about="⟨uri⟩"/>
              | <Class>
                  ⟨classconst⟩?
                  ⟨classcond⟩ *
                </Class>
```

So a ⟨*classexpr*⟩ is either a class identified by its URI or a complex class expression made of restrictions and/or constructions. Here, the conditons of the abstract syntax are directly associated with the other connectors. Construction for classes are the usual boolean connectives:

```
⟨classconst⟩ ::= <and>
                   <Collection> (<item>⟨classexpr⟩</item>)+ </Collection>
                 </and>
               | <or>
                   <Collection> (<item>⟨classexpr⟩</item>)+ </Collection>
                 </or>
               | <not>⟨classexpr⟩</not>
```

The following example shows a correspondence featuring a class expression using the ”or” constructor.

```
<align:Cell>
  <align:entity1>
    <Class rdf:about="&wine;WineFlavor"/>
  </align:entity1>
  <align:entity2>
    <Class>
      <or><Collection>
        <item><Class rdf:about="&vin;Acidite"/></item>
        <item><Class rdf:about="&vin;Astreingence"/></item>
        <item><Class rdf:about="&vin;Amertume"/></item>
      </Collection></or>
    </Class>
  </align:entity2>
  <align:measure rdf:datatype="&xsd;float">1.0</align:measure>
  <align:relation>subsumes</align:relation>
</align:Cell>
```

Conditions are constraints applying to classes:

$\langle$*classcond*$\rangle$ ::= `<attributeValueCondition>` $\langle$*constraint*$\rangle$ `</attributeValueCondition>`
| `<attributeTypeCondition>` $\langle$*constraint*$\rangle$ `</attributeTypeCondition>`
| `<attributeOccurenceCondition>` $\langle$*constraint*$\rangle$ `</attributeOccurenceCondition>`

These three types of conditions restrict the scope of a class expression by applying a restriction to the value, type or cardinality of a particular property or relation. They enforce the presence of a particular attribute, the class of the object instantiating the attribute or the value of this attribute in the case of a datatype.

```
⟨constraint⟩ ::= <Restriction>
                   <onProperty>⟨path⟩</onProperty>
                   <comparator>⟨uri⟩</comparator>
                   <value>⟨pov⟩</value>
                 </Restriction>
```

The attribute on which the condition is based is indicated using a path. A path is used to target a specific entity in the graph made by classes, properties and relations. It allows, for exemple, that a correspondence only applies to instances which are involved in a particular relation: "Wines whose producing area is located in a particular region". The following example[2] expresses this path:

```
<Path>
  <first><Relation rdf:about="&vin;hasTerroir"/></first>
  <next><Relation rdf:about="&proton;LocatedIn"/></next>
</Path>
```

Here is the syntax for a path:

```
⟨path⟩    ::= <Property rdf:about="⟨uri⟩"/>
            | <Relation rdf:about="⟨uri⟩"/>
            | <Self />
            | <Path>
                <first>⟨step⟩</first>
                <next>⟨path⟩</next>
              </Path>

⟨step⟩    ::= <Relation rdf:about="⟨uri⟩"/>
```

This syntax does not prohibits through the grammar to have several attribute steps in a path. As we have seen before, paths can be used in place of values, the $\langle$*pov*$\rangle$ production (for path-or-value expresses this):

```
⟨value⟩   ::= ⟨string⟩
            | ⟨instance⟩
            | <Apply operation=" ⟨uri⟩ "> ⟨pov⟩* </Apply>

⟨pov⟩     ::= ⟨path⟩
            | ⟨value⟩
```

[2]This example makes use of the Proton ontology: http://proton.semanticweb.org/.

Thanks to paths and restrictions, it is possible to restrict a correspondence to “Wines whose producing region is located in Aquitaine” using the following restriction. The comparator is issued from the comparators list given table C.2 in Appendix.

```
<Restriction>
  <onProperty>
    <Path>
      <first><Relation rdf:about="&vin;hasTerroir"/></first>
      <next><Relation rdf:about="&proton;LocatedIn"/></next>
    </Path>
  </onProperty>
  <comparator rdf:resource="equal"/>
  <value rdf:datatype="&xsd;string">Acquitaine</value>
</Restriction>
```

We can with this restriction build the correspondence from § 46.2 between “BordeauxWine” and “Vin” whose “terroir” is located in “Aquitaine”[3].

```
<align:Cell>
  <align:entity1>
    <Class rdf:about="&wine;BordeauxWine">
  </align:entity1>
  <align:entity2>
    <Class rdf:about="&vin;Vin">
      <attributeValueCondition>
        <Restriction>
          <onProperty>
            <Path>
              <first><Relation rdf:about="&vin;hasTerroir"/></first>
              <next><Relation rdf:about="&proton;LocatedIn"/></next>
            </Path>
          </onProperty>
          <comparator rdf:resource="equal"/>
          <value rdf:datatype="&xsd;string">Acquitaine</value>
        </Restriction>
      </attributeValueCondition>
    </Class>
  </align:entity2>
</align:Cell>
```

An example of occurrence restriction would be the wines produced in a region with no adjacent region, such as an island. For instance Moscatel Madeira wine is produced on the island of Madeira.

```
<align:Cell>
  <align:entity1>
    <Class rdf:about="&wine;Wine">
      <attributeOccurrenceCondition>
        <Restriction>
          <onProperty>
            <Path>
              <first><Relation rdf:about="&wine;locatedIn"/></first>
              <next>< Relation rdf:about="&wine;adjacentRegion"/></next>
```

[3]Aquitaine is the administrative region to which belongs the city of Bordeaux

```
          </Path>
        </onProperty>
        <comparator rdf:resource="cardinality"/>
        <value rdf:datatype="&xsd;int">0</value>
      </Restriction>
    </attributeOccurrenceCondition>
  </Class>
</align:entity1>
<align:entity2>
  <Class rdf:about="&vin;Madere" />
</align:entity2>
<align:measure rdf:datatype="&xsd;float">1.0</align:measure>
<align:relation>subsumes</align:relation>
</align:Cell>
```

Another example modeling the correspondence presented in the scenario § 46.2. This correspondence uses a path to state that a locally grown wine is a wine whose owner (propriétaire) is the same person as the first seller (négociant) for this wine.

```
<align:Cell>
  <align:entity1>
    <Class rdf:about="&wine;LocallyGrownWine"/>
  </align:entity1>
  <align:entity2>
    <Class rdf:about="&vin;Vin">
      <attributeValueCondition>
        <Restriction>
          <onProperty>
            <Relation rdf:about="&vin;propriétaire"/></first>
          </onProperty>
          <comparator rdf:resource="equals"/>
          <value><Relation rdf:about="&vin;négotiant"/></value>
        </Restriction>
      </attributeValueCondition>
    </Class>
  </align:entity2>
  <align:measure rdf:datatype="&xsd;float">1.0</align:measure>
  <align:relation>subsumes</align:relation>
</align:Cell>
```

51.1.3 Relation expression

Relation expressions allow constructing relations using a set of operators. Complementary to the operators described in class expressions, specific relation operators are here introduced. The ⟨*inverse*⟩ operator take the inverse relation of a given one. The ⟨*symmetric*⟩, ⟨*transitive*⟩ and ⟨*reflexive*⟩ operators construct the symmetric, transitive and reflexive closure of a given relation. Semantics for these operators are given in § 50.3.8.

Relation expressions can be either a relation from an ontology, a construction from such relations or constraints on these relations:

```
⟨relexpr⟩  ::= <Relation rdf:about="⟨uri⟩" />
             | <Relation>
                 ⟨relconst⟩?
```

```
    ⟨relcond⟩ *
    ⟨transformation⟩?
  </Relation>
```

The constructors for relations are those of relation algebra. These are the boolean operators (and, or and not) and the binary relation operators (converse, symmetric closure, transitive closure and reflexive closure).

```
⟨relconst⟩ ::= <and>
                 <Collection> (<item>⟨classexpr⟩</item>)+ </Collection>
               </and>
             | <or>
                 <Collection> (<item>⟨classexpr⟩</item>)+ </Collection>
               </or>
             | <not>⟨relexpr⟩</not>
             | <inverse>⟨relexpr⟩</inverse>
             | <symmetric>⟨relexpr⟩</symmetric>
             | <transitive>⟨relexpr⟩</transitive>
             | <reflexive>⟨relexpr⟩</reflexive>
```

The restrictions are the usual domain and range restrictions:

```
⟨relcond⟩ ::= <domainRestriction>⟨classexpr⟩</domainRestriction>
            | <rangeRestriction>⟨classexpr⟩</rangeRestriction>
```

Domain restriction is exemplified in the following relation expression:

```
<align:Cell>
  <align:entity1>
    <Relation rdf:about="&wine;locatedIn">
      <domainRestriction>
        <Class rdf:about="&wine;Wine" />
      </domainRestriction>
    </Relation>
  </align:entity1>
  <align:entity2>
    <Relation rdf:about="&vin;has_terroir" />
  </align:entity2>
  <align:measure rdf:datatype="&xsd;float">1.0</align:measure>
  <align:relation>subsumedBy</align:relation>
</align:Cell>
```

51.1.4 Property expression

As class expressions, property expressions give the possibility to group properties using constructors. We have divided them into Properties and Relations according to the abstract syntax.

Properties can either be directly addressed through their URI, constructed, restricted and transformed.

```
⟨propexpr⟩ ::= <Property rdf:about="⟨uri⟩" />
             | <Property>
                 ⟨attrconst⟩?
                 ⟨attrcond⟩*
               </Property>
```

The Property constructors are the usual boolean constructors:

```
⟨attrconst⟩ ::= <and>
                 <Collection> (<item>⟨propexpr⟩</item>)+ </Collection>
                </and>
              | <or>
                 <Collection> (<item>⟨propexpr⟩</item>)+ </Collection>
                </or>
              | <not>⟨propexpr⟩</not>
```

The Property constraints are the domain and range restriction. The range restriction is specified through constraints on the types and values of the range of the property:

```
⟨attrcond⟩ ::= <domainRestriction>⟨classexpr⟩</domainRestriction>
             | <typeRestriction>⟨datatype⟩</typeRestriction>
```

51.1.5 Attribute expression

Attribute expressions are a restriction of a property in the context of a particular class:

```
⟨attexpr⟩  ::= <Attribute>
                 <context> ⟨classexpr⟩ </context>
                 <onProperty> ⟨relexpr⟩</onProperty>
               </Attribute>
```

51.1.6 Instances

Instance expressions are simply specified using the instance URIs:

```
⟨instance⟩ ::= <Instance rdf:about="⟨uri⟩"/>
```

51.1.7 Metadata

It is often useful to annotate alignments with additional information. Metadata gives the opportunity to programs to exchange information which is not part of the format. This was not part of the abstract syntax.

This is achieved by introducing annotations in Alignment and Cell expressions. These annotations contain string values.

⟨*annotation*⟩ ::= < ⟨*uri*⟩ > ⟨*annotationValue*⟩ </ ⟨*uri*⟩ >

⟨*annotationValue*⟩ ::= ⟨*string*⟩

There are a number of common annotations that can be used already. They are summarised in Table 51.1[4].

The set of annotations can be extended by the users and it is a good practice for tools to be able to preserve all these annotations.

[4]Other types of declared annotations can be found at `http://alignapi.gforge.inria.fr/labels.html`

Annotation	Type	Content
type	char	the kind of alignment it is (1:1 or n:m for instance)
level	xsd:string	the language level used in the alignment (level 0 for the initial alignment API, level 2OMWG for the language defined here)
method	classname	the algorithm that provided it (or if it has been provided by hand)
dc:creator	xsd:string/URI	the person who produced the alignment
dc:date	xsd:date	the date of creation or modification for the alignment
purpose	xsd:string	the purpose for which the alignment has been produced
parameters	Parameters	the parameters passed to the generating algorithm
time	xsd:duration	the time spent for generating the alignment
limitations	xsd:string	the limitations of the use of the alignment
properties	undef	the properties satisfied by the correspondences (and their proof if necessary)
certificate	undef	the certificate from an issuing source
arguments	Arguments	the arguments in favour or against a correspondence

Table 51.1: Metadata labels for annotating alignments.

51.2 Surface syntax

The surface syntax is meant to be a human readable syntax. It is more compact than the XML/RDF exchange syntax but more explicit than the abstract syntax. Its compactness helps providing a short description of it through a grammar. This language is an evolution of the surface language described in [Scharffe and Kiryakov, 2005].

An alignment is expressed as a mapping document which has an identifier, source and target ontology references, possibly annotations and a list of correspondences (⟨*expression*⟩):

⟨*mappingdocument*⟩ ::= `MappingDocument (` ⟨*documentid*⟩
⟨*headers*⟩
⟨*sourceexp*⟩
⟨*targetexp*⟩
⟨*annotation*⟩*
⟨*expression*⟩* `)`

⟨*header*⟩ ::= (⟨*entity*⟩ | ⟨*namespace*⟩)*

⟨*entity*⟩ ::= `XMLentity(` ⟨*string*⟩ ⟨*irid*⟩ `)`

⟨*namespace*⟩ ::= `namespace(` ⟨*string*⟩ ⟨*irid*⟩ `)`

⟨*sourceexp*⟩ ::= `onto1(` ⟨*formalism*⟩ ⟨*ontologyid*⟩ `)`

⟨*targetexp* ⟩ ::= `onto2(` ⟨*formalism*⟩ ⟨*ontologyid*⟩ `)`

⟨*annotation*⟩ ::= `annotation(` ⟨*irid*⟩ ⟨*propertyvalue*⟩ `)`

⟨*formalism*⟩ ::= `formalism(` ⟨*string*⟩ ⟨*irid*⟩ `)`

Annotations are property-values pairs; ontologies are simply described by their identifier and language description.

⟨*expression*⟩ ::= `MappingRule(` ⟨*mappingid*⟩ ⟨*annotation*⟩* ⟨*measure*⟩? ⟨*relation*⟩?
`entity1(` ⟨*entity*⟩ `)` `entity2(` ⟨*entity*⟩ `)`

⟨*mappingid*⟩ ::= `id(` ⟨*irid*⟩ `)`

⟨*measure*⟩ ::= `measure(` ⟨*float*⟩ `)`

⟨*relation*⟩ ::= `relation(` ⟨*relname*⟩ `)`

⟨*relname*⟩ ::= `equivalent` | `subsume` | `subsumed` | `isa` | `asi`

The entities which are put in correspondences are either classes, relations, properties, attributes or instance expressions. They can be associated with transformations which are either internal functions or service call:

⟨*entity*⟩ ::= `Class(` ⟨*classexpr*⟩ `)`
| `Relation(` ⟨*relationexpr*⟩ `)`
| `Property(` ⟨*propertyexpr*⟩ `)`
| `Attribute(` ⟨*classexpr*⟩ ⟨*relationexpr*⟩ `)`
| `Instance(` ⟨*instanceid*⟩ `)`

Class, relation and attribute expressions closely corresponds to those presented in the previous sections:

⟨*classexpr*⟩ ::= ⟨*classid*⟩
| `and(` [first]: ⟨*classexpr*⟩ [second]: ⟨*classexpr*⟩+ `)`
| `or(` [first]: ⟨*classexpr*⟩ [second]: ⟨*classexpr*⟩+ `)`
| `not(` ⟨*classexpr*⟩ `)`
| ⟨*condition*⟩

⟨*relationexpr*⟩ ::= ⟨*relationid*⟩
| `and(` [first]: ⟨*relationexpr*⟩ [second]: ⟨*relationexpr*⟩+ `)`
| `or(` [first]: ⟨*relationexpr*⟩ [second]: ⟨*relationexpr*⟩+ `)`
| `not(` ⟨*relationexpr*⟩ `)`
| `domain(` ⟨*classexpr*⟩ `)`
| `range(` ⟨*classexpr*⟩ `)`
| `inverse(` ⟨*relationexpr*⟩ `)`
| `symetric(` ⟨*relationexpr*⟩ `)`
| `transitive(` ⟨*relationexpr*⟩ `)`
| `reflexive(` ⟨*relationexpr*⟩ `)`

⟨*propertyexpr*⟩ ::= ⟨*propertyid*⟩
| `and(` [first]: ⟨*propertyexpr*⟩ [second]: ⟨*propertyexpr*⟩ ⟨*propertyexpr*⟩* `)`
| `or(` [first]: ⟨*propertyexpr*⟩ [second]: ⟨*propertyexpr*⟩ ⟨*propertyexpr*⟩* `)`
| `not(` ⟨*propertyexpr*⟩ `)`
| `domain(` ⟨*classexpr*⟩ `)`
| `range(` ⟨*typeexpr*⟩ `)`

Conditions can be specified on paths so these are introduced and defined precisely as in the previous sections:

⟨*condition*⟩ ::= `valuerestriction(` ⟨*path*⟩ ⟨*comparator*⟩ ⟨*pathorvalue*⟩ `)`
  | `domainrestriction(` ⟨*path*⟩ ⟨*comparator*⟩ ⟨*typeid*⟩ `)`
  | `cardinality(` ⟨*path*⟩ ⟨*comparator*⟩ ⟨*number*⟩ `)`

⟨*comparator*⟩ ::= `>` | `>=` | `<` | `<=` | `=` | `!=` | ⟨*irid*⟩

⟨*pathorvalue*⟩ ::= ⟨*path*⟩
  | ⟨*literal*⟩
  | `instance(` ⟨*instanceid*⟩ `)`
  | ⟨*transformation*⟩

⟨*path*⟩ ::= ⟨*relationid*⟩
  | ⟨*propertyid*⟩
  | `path(` ⟨*relationid*⟩* ⟨*propertyid*⟩? `)`

⟨*transformation*⟩ ::= `transformation(` ⟨*functionid*⟩ ⟨*pathorvalue*⟩* `)`
  | `transformation(` ⟨*service*⟩ ⟨*irid*⟩ ⟨*pathorvalue*⟩* `)`

⟨*functionid*⟩ ::= ⟨*string*⟩ | ⟨*irid*⟩

There are two different ways to express transformations depending on their reliance on embedded functions or web services.

Finally litterals are identified like in RDF

⟨*literal*⟩ ::= ⟨*typedliteral*⟩
  | ⟨*plainliteral*⟩
  | ⟨*number*⟩
  | ⟨*string*⟩

⟨*typedliteral*⟩ ::= ⟨*plainliteral*⟩ `^^` ⟨*typeid*⟩

⟨*plainliteral*⟩ ::= `"` ⟨*literalcontent*⟩* `"` ⟨*languagetag*⟩?

Most items are identified by IRIs (the internationalised version of URIs, i.e., allowing other characters than ASCII).

⟨*documentid*⟩ ::= ⟨*irid*⟩

⟨*ontologyid*⟩ ::= ⟨*irid*⟩

⟨*classid*⟩ ::= ⟨*irid*⟩

⟨*propertyid*⟩ ::= ⟨*irid*⟩

⟨*attributeid*⟩ ::= ⟨*irid*⟩

⟨*relationid*⟩ ::= ⟨*irid*⟩

⟨*instanceid*⟩ ::= ⟨*irid*⟩

⟨*irid*⟩ ::= <" ⟨*iri*⟩ ">

Here is an example of this surface syntax:

```
MappingDocument(
  id(<"http://oms.omwg.org/example/">)
  XMLEntity("wine" <"http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#">)
  XMLEntity("vin" <"http://ontology.deri.org/vin#">)
  namespace("xmlns" <"http://knowledgeweb.semanticweb.org/heterogeneity/alignment">)
  namespace("xmlns:dc" <"http://purl.org/dc/elements/1.1/">)
  namespace("xmlns:align" <"http://www.w3.org/2001/XMLSchema#">)
  annotation(<"dc:creator"> "http://sw.deri.org/˜francois/")
  annotation(<"dc:date"> "2006/06/07")
  annotation(<"method">, "manual")
  annotation(<"level">, "2OMWG")
  annotation(<"type">, "**")
  onto1( formalism('owl',<"http://www.w3.org/2002/07/owl">) <"&wine;"> )
  onto2( formalism('wsml',<"http://www.wsmo.org/wsml/wsml-syntax/wsml-dl">) <"&vin;"> )
  MappingRule(
       id("MappingRule_0")
       measure(1.0)
       relation(equivalence)
       entity1( Class(<"&wine;VintageYear">))
       entity2( Class(<"&vin;Millesime">)))
  MappingRule(
       id("MappingRule_1")
       measure(1.0)
       relation(equivalence)
       entity1( Attribute(<"&wine;yearValue">))
       entity2( Attribute(<"&vin;anneeMillesime">)))
  MappingRule(
       id("MappingRule_2")
       measure(.9)
       relation(subsumption)
       entity1( Relation(<"&wine;locatedIn">))
       entity2( Relation(<"&vin;hasTerroir">)))
  MappingRule(
       id("MappingRule_3")
       measure(1.0)
       relation(subsumption)
       entity1( Class(<"&wine;Bordeaux">))
       entity2( Class(and(<"&vin;Vin">,
                          valuerestriction(
                              path(<"vin;hasTerroir"> <"&proton;LocatedIn">)
                              equal "Bordeaux")))))
  MappingRule(
       id("MappingRule_5")
       measure(1.0)
       relation(equivalence)
       entity1( Property(<"&wine;hasVintageYear">))
       entity2(Property(and(<"&vin;annee_millesime">
                        transformation(Transformation:Date2Int))))))
  MappingRule(
       id("MappingRule_4")
       measure(1.0)
       relation(subsumes)
       entity1( Class(<"&vin;WineFlavor">))
       entity2(Class(or(<"&vin;Acidite">
                        <"&vin;Astreingence">
                        <"&vin;Amertume">)))))
```

51.3 Synthesis

We have provided two different syntax for expressive alignments. They are expressive enough for covering the examples provided in Chapter 54. These syntaxes are designed for being strictly equivalent, though (for space reasons) we did not provide transformation rules. The exchange syntax and the surface syntax are designed for being realised and handled by systems. They are indeed handled in input and output by the OMWG parsers. The abstract syntax is only meant for specifying the semantics of such alignments and is not available in any tool.

Going one step further in this direction would be the design of a graphical expressive syntax for alignments. However, it is difficult to separate such a syntax from ontology syntax. The figures of Chapter 54 could be considered as the sketch of such a syntax and they display a syntax for ontology languages as well.

Next chapter consider the implementation of this expressive language.

Chapter 52

Expressive language implementation

We have implemented tools for supporting the new alignment format by extending and using together two ontology alignment tools. These two tools where developed relatively independently but in compatible ways so we have adapted them in order to achieve the best implementation of the proposed language. We first present both tools independently and explain the integration that has been achieved.

52.1 OMWG Mapping API

The OMWG Mapping API[1] has been developed for providing an expressive language to mediate between semantic web services in the WSMX framework. However, it is independent from that framework. This section introduce the Java API developed and implemented to support the mapping language. The different component constituting the API, namely the parsers, the object model, the export module and the adapter interface are presented.

52.1.1 Parsers

The mapping language has two syntaxes: a surface syntax and a RDF/XML syntax.

The surface syntax parser is based on an EBNF grammar and corresponds to the language presented in §51.2. We used Sablecc[2] to generate a parser from the grammar specification. A tree-walker goes through the abstract-syntax tree and populate the object model of the API. The EBNF grammar is available in [Scharffe and Kiryakov, 2005].

The RDF/XML syntax parser is based on the javax.xml.parser[3] Java package and works similarly.

52.1.2 Object Model

The parsers instanciate an object model providing means to manipulate alignments. Classes and properties of the model follow the structure of the language. The top-level class MappingDocument contains information about the document: the source ontology, the target ontology, the different correspondences and annotations. It is possible to access and modify annotations, id of

[1] http://sourceforge.net/projects/mediation/
[2] http://www.sablecc.org
[3] http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/parsers/package-summary.html

the document, the source and target ontologies. It is also possible to add or suppress a correspondence. Once the mapping document edited, it can be exported via the export module.

52.1.3 Export Module

The export module offers the possibility to export mapping documents in various gounding formats. It allows to export in the surface syntax of the mapping language, in OWL and in WSML. This implementation gives the possibility to use alignments at run-time by loading them into a mediator.

52.1.4 Known uses

The mapping API is used by the mapping editor from the Web Services Modelling Toolkit[4] (WSMT) in order to represent ontology alignments and execute them in a runtime mediator. In a similar way, Ontomap, the mapping tool from OntoStudio[5] allows to export alignments as mapping documents using the Mapping API. An online store for mapping documents[6] makes use of the Mapping API to check the validity of submitted alignments.

52.2 Alignment API implementation

The Alignment API[7] is an API and implementation for expressing and sharing ontology alignments [Euzenat, 2004]. It is a Java description of tools for accessing the common format. It defines four main interfaces (Alignment, Cell, Relation and Evaluator) and proposes the following services:

– Storing, finding, and sharing alignments;
– Piping alignment algorithms (improving an existing alignment);
– Manipulating (thresholding and hardening);
– Generating processing output (transformations, axioms, rules);
– Comparing alignments.

52.2.1 Parsers

The Alignment API uses a general format for expressing alignments in a uniform way. The goal of this format is to be able to share on the web the available alignments. It helps systems using alignments, e.g., mergers, translators, to take advantage of any alignment algorithm and it will help alignment algorithms to be used in many different tasks. The format is expressed in RDF, so it is freely extensible, and has been defined by a DTD (for RDF/XML), an OWL ontology and an RDF Schema. Aligned entities are identified by their URIs.

However, the parser is designed to work only with the XML format.

[4]http://wsmt.sourceforge.net
[5]http://www.ontoprise.com
[6]http://oms.omwg.org
[7]http://alignapi.gforge.inria.fr

52.2.2 Object model and manipulation

The object model implements the Alignment API and provides the following features:

- a base implementation of the interfaces with all useful facilities;
- a library of sample matchers;
- a library of renderers;
- a library of evaluators (precision/recall, generalized precision/recall, precision/recall graphs and weighted Hamming distance);
- a parser for the format.

This implementation is now made available as an Alignment server which offers all these functions through HTTP/HTML, HTTP/SOAP and FIPA ACL interfaces.

Instanciating this API is achieved by refining the base implementation by implementing the align() method. Doing so, the new implementation will benefit from all the services already implemented in the base implementation.

52.2.3 Renderers

Renderers corresponding to the output formats are available for XSLT, SWRL, OWL, C-OWL, OMWG Mapping Language, and SKOS.

52.2.4 Known uses

The alignment API has been used for the processing of the EON Ontology Alignment Contest and the Ontology Alignment Evaluation Initiative 2005. It is used in the people's portal alignment tool at DERI Innsbruck and used or output by a number of alignment tools (among which OLA that we develop in common with the University of Montral, CMS from University of Southampton or oMap from CNR/Pisa).

52.3 Combination of both API

Because these two tools are currently in use under different context we aimed at integrating them by preserving their independent behaviour. The goal of integrating these tools was to have the Alignment API benefit from an already expressive alignment language and maybe the WSMT alignment editor. From the standpoint of the OMWG Mapping API, being sited on top of the Alignment API provides a number of facilities: being used in all the contexts in which the Alignment API is used (in particular various matching algorithms and the Alignment server) and providing low level facilities (more renderers, sophisticated extraction algorithms, etc.).

Because the two implementations are compatible, we implemented them by making the OMWG mapping API object model an extension of the Alignment API implementation object language. Then, we "reimplemented" the base accessors of the OMWG mapping API to the accessors of the Alignment API so that any tool using the Alignment API could manipulate the MappingDocuments and the correspondences they contain. This is presented in Figure 52.1.

Parsing the Mapping Documents is something that was fully implemented in the OMWG mapping API, so we made the Alignement API to take advantage of them. Concerning the surface language, the corresponding OMWG parser, when invoked, creates MappingDocuments which

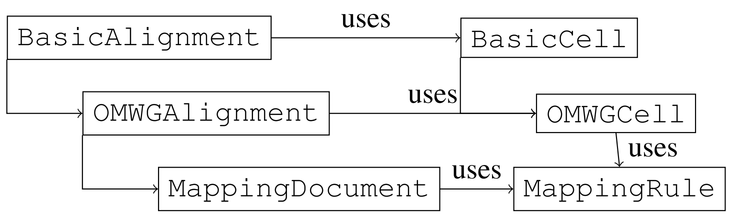


Figure 52.1: Integration of the object model of the OMWG Mapping API on top of the Alignment API.

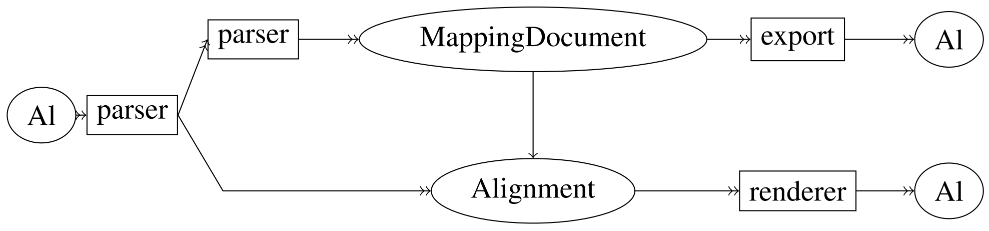


Figure 52.2: Invocation of the functions of the OMWG Mapping API and the Alignment API on the resulting objects.

are now also Alignment in the sense of the Alignment API, so they can be manipulated by this API. Concerning the RDF/XML format, we make the Alignment API parser switch to the OMWG mapping API parser whenever it encounters an alignment written in this expressive language. This is achieved when the parser recognises the ”2OMWG” value in the level attribute of the alignment. The result is also a MappingDocument.

The objects created by the two parsers are now objects of the Alignment API and can be manipulated as such. Since they are still objects of the OMWG mapping API, then they can also be used as such.

Last, the Alignment API has a particular way of outputing alignments through “renderers”. These had to be extended to be used with MappingDocuments. The Mapping API export module is still usable with the MappingDocuments (and the RDF/XML renderer of the Alignment API reuses parts of the OMWG implementation).

These dispositions are presented in Figure 52.2.

52.4 Synthesis

We have presented the integration of two existing tools in order to implement support for the expressive alignment language that has been presented in this deliverable. The result is fully integrated within both the Alignment API implementation and the OMWG Mapping API. It can be found at the Alignment API URL for the moment (starting with version 3.1).

This integration allows to retain the advantage of both tools, including the expressive language described here, while benefiting of the advantage of the other.

Chapter 53

Conclusions

The ontology alignment syntax and semantics presented in this report form a complete language. It is possible to use this language for modelling complex correspondences between ontologies. As shown on simple examples between two ontologies related to the wine domain, such complex correspondences are necessary to correctly represent the domain overlap between two heterogeneous ontological representations.

We provided support for this language in terms of expressing correspondences and manipulating them in a practical already usable setting. By combining the simplicity of the Alignment Format and the expressivity of the Mapping Language, we propose a language fulfilling the need for ontology mediation on the semantic web. The presented language takes into account the different types of entities used to represent structured data and allows as well to specify transformation of the data itself. Moreover, this language enables mediation between heterogeneous ontologies or schemata described using different languages. Mediating between different formalisms is made possible by the use of model theoretic semantics using local interpretations lifted to a global domain via equalising functions.

We provided an alignment language as expressive as possible without any regard to its complexity. We aimed at providing a very expressive language that people will taylor if they do not need the expressiveness or if they want to trade it for decidable or efficient reasoning. By not restricting the expressiveness of the language a priori, we have allowed ourselves to provide the syntax and the semantics for the whole language at once. In fact, reasoning with such languages requires not uniquely reasoners in this language but reasoners in the two connected ontology languages. Of course, providing inference support for such a language could be an interesting challenge.

The language as described will not look any new for someone moderately accointed with description logics [Baader *et al.*, 2003]. Indeed, we did not attempt to be original in this matter. But the reader should note that this language allows to use any kind of language as ontology languages. It is not meant to be used only with ontology languages based on description logics, though we assume a model theoretic semantics to these languages.

This language can be used for many different applications. From graphical user interfaces assisting in aligning ontologies to web service mediator rewriting queries or transforming instances a wide range of application can be foreseen. The expressiveness of this language might lead research in ontology matching to develop algorithms able to find complex correspondences such as those presented here.

Part VII

Processing alignments

Chapter 54

Introduction

In Knowledge web, we have taken a two steps view on reducing semantic heterogeneity: (i) matching entities to determine alignments and (ii) processing alignments according to application needs.

So far, we more specifically investigated the first step through defining alignments (D2.2.1), proposing alignment formats (D2.2.6, D2.2.10) and semantics (D2.2.5) and evaluating matching results (D2.2.2, D2.2.4, D2.2.9). Many Knowledge web partners have produced ontology matching systems.

In this deliverable, we present how the alignments can be specifically used by applications, thus focusing on the alignment processing step. We also consider the need for considering alignments in the long term, i.e., alignment management. We present the broad classes of alignment use and the tools for implementing these usages.

The remainder of this introduction is as follows: first we recall the kinds of applications which need matching and we elicit their requirements about the matching operation and the use of alignements (§54.1); then, we illustrate the use of alignment processing in data mediation (§54.2); finally, we replace the ontology matching operation within the broader view of what we call the alignment life cycle (§54.3).

Some parts of this deliverable have been published in [Euzenat and Shvaiko, 2007] and [Euzenat *et al.*, 2008].

54.1 Applications

Several classes of applications can be considered (they are more extensively described in [Euzenat and Shvaiko, 2007], we only summarise them here). They are the following:

Ontology evolution uses matching for finding the changes that have occurred between two ontology versions [De Leenheer and Mens, 2008].

Schema integration uses matching for integrating the schemas of different databases under a single view;

Catalog integration uses matching for offering an integrated access to on-line catalogs;

Data integration uses matching for integrating the content of different databases under a single database;

P2P information sharing uses matching for finding the relations between ontologies used by different peers;

Web service composition uses matching between ontologies describing service interfaces in order to compose web services by connecting their interfaces;

Multiagent communication uses matching for finding the relations between the ontologies used by two agents and translating the messages they exchange;

Context matching in ambient computing uses matching of application needs and context information when applications and devices have been developed independently and use different ontologies;

Query answering uses ontology matching for translating user queries between heterogeneous data sources on the web.

Semantic web browsing uses matching for dynamically (while browsing) annotating web pages with partially overlapping ontologies.

It is clear, from the above examples, that matching ontologies is a major issue in ontology related activities. It is not circumscribed to one application area, but applies to any application that communicates through ontologies.

These kinds of applications have been analysed in order to establish their requirements with regard to matching systems. The most important requirements concern:

– the type of available input a matching system can rely on, such as schema or instance information. There are cases when data instances are not available, for instance due to security reasons or when there are no instances given beforehand. Therefore, these applications require only a matching solution able to work without instances (here a schema-based method).
– some specific behaviour of matching, such as requirements of (i) being automatic, i.e., not relying on user feed-back; (ii) being correct, i.e., not delivering incorrect matches; (iii) being complete, i.e., delivering all the matches; and (iv) being performed at run time.
– the use of the matching result as described above. In particular, how the identified alignment is going to be processed, e.g., by merging the data or conceptual models under consideration or by translating data instances among them.

In particular, there is an important difference between applications that need alignments at design time and those that need alignments at run time. Ontology evolution is typically used at design time for transforming an existing ontology which may have instances available. It requires an accurate, i.e., correct and complete, matching, but can be performed with the help of users. Schema, catalogue and data integration are also performed off-line but can be used for different purposes: translating data from one repository to another, merging two databases or generating a mediator that will be used for answering queries. They also will be supervised by a human user and can provide instances. Other applications are rather performed at run time. Some of these, like P2P information sharing, query answering and semantic web browsing are achieved in presence of users who can support the process. They are also less demanding in terms of correctness and completeness because the user will directly sort out the results. On the other hand, web-service composition, multiagent communication and context matching in ambient computing

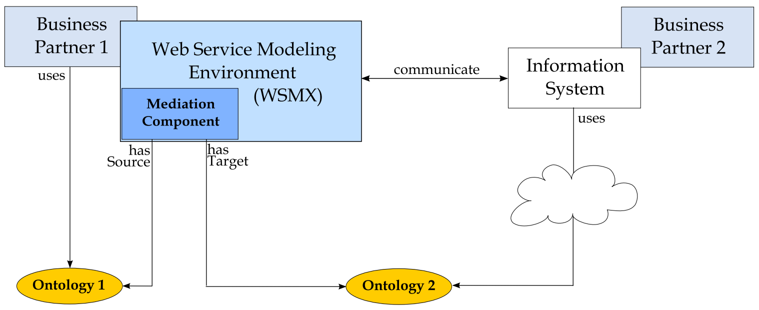


Figure 54.1: Instance transformation scenario.

require matching to be performed automatically without assistance of a human being. Since, the systems will use the result of matching for performing some action (mediating or translating data) which will be fed in other processes, correctness is required. Moreover, usually these applications do not have instance data available.

54.2 Example: data mediation for semantic web services

Web services represent one of the areas where data mediation is the most required. Services are resources usually developed independently which greatly vary from one provider to another in terms of the used data formats and representation. By adding semantics to web services, heterogeneity problems do not disappear but require more intelligent dynamic and flexible mediation solutions. Ontologies which carry most of these explicit semantics become the crucial elements to support the identification and capturing of semantic mismatches between models.

Web Services Execution Environment (WSMX) is a framework that enables discovery, selection, invocation and interoperation of Semantic Web services [Mocan *et al.*, 2006]. Ontology-based data mediation plays a crucial role in enabling all the above mentioned service operations. Different business actors use ontologies to describe their services internal business logic, and, more importantly in this case, their data. Each of these actors uses its own information system, e.g., WSMX, and tries to interact with other actors, part of other (probably more complex) business processes (Figure 54.1). A specialised component or service is needed to transform the data expressed in terms of a given ontology (the source ontology) in the terms of another ontology (target ontology), allowing the two actors to continue using their own data representation formats. Being part of a run time process the data, i.e., instances, transformation has to be performed completely automatically. In addition, as soon as such a mediator has to act in a business environment, the result of the mediation process has to be correct and complete at all time.

In order to achieve these three requirements (automation, correctness and completeness), the whole process is split in two phases: a design time phase which covers the correctness and completeness by involving the human domain expert and the run time phase when the mediation is performed in an automatic manner based on the alignments established at design time.

We will provide further details on these two phases in Section 55 and Section 59; Section 61 will consider the management of the alignments between these two phases.

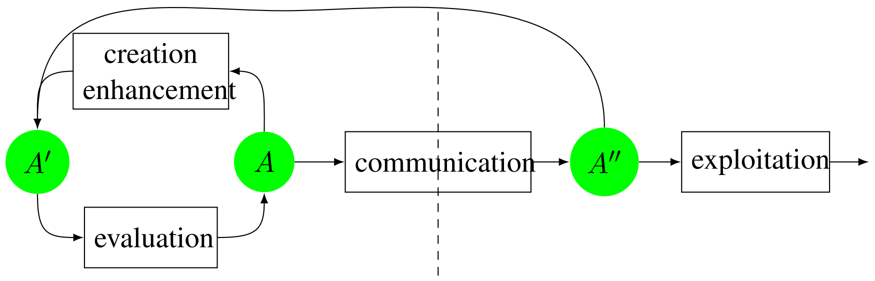


Figure 54.2: The ontology alignment life cycle.

54.3 The alignment life cycle

As the example shows, the difference between design time and run time is very relevant to ontology management. On the one hand, if alignments are required at design time, then ontology developers will need support in creating, manipulating and using these alignments. On the other hand, if alignments are required at run time, then one way of ensuring timely and adequate response may be to find some existing alignment in an alignment store. Alignments stored there should be carefully evaluated and certified alignments. They thus require alignment management on their own.

Like ontologies, alignments have their own life cycle (see Figure 54.2) and users should be supported in manipulating alignments during their life cycle. They are first created through a matching process (which may be manual). Then they can go through an iterative loop of evaluation and enhancement. Again, evaluation can be performed either manually or automatically, it consists of assessing properties of the obtained alignment. Enhancement can be obtained either through manual change of the alignment or application of refinement procedures, e.g., selecting some correspondences by applying thresholds. When an alignment is deemed worth publishing, then it can be stored and communicated to other parties interested in it. At this stage, users feedback might lead to modification of the alignment. Finally, the alignment is transformed into another form or interpreted for performing actions like mediation or merging. This last step is the specific topic of this deliverable. However, it cannot be considered independently of the previous steps.

To this first independent cycle is added the joint life cycle that can tie ontologies and alignments. As soon as ontologies evolve, new alignments have to be produced for following this evolution. This can be achieved by recording the changes made to ontologies and transforming these changes into an alignment (from one ontology version to the next one). This can be used for computing new alignments that will update the previous ones. In this case, previously existing alignments can be replaced by the composition of themselves with the ontology update alignment (see Figure 54.3).

Taking seriously ontology management requires to involve alignment management with ontology management. However, so far very few tools offer support for alignment management, let alone, joint ontology-alignment support.

$$\begin{tikzcd}
d \arrow[r, two heads] \arrow[d] & o \arrow[r] \arrow[d] & A & o' \arrow[l] \arrow[dl] \\
T \arrow[d] & A' \arrow[l] & A \cdot A' & \\
d' \arrow[r, two heads] & o_1 \arrow[u] \arrow[ur] & &
\end{tikzcd}$$

Figure 54.3: Evolution of alignments. When an ontology o evolves into a new version o_1, it is necessary to update the instances of this ontology (d) and the alignments (A) it has with other ontologies (o'). To that extent, a new alignment (A') between the two versions can be established and it can be used for generating the necessary instance transformation (T) and updated alignments ($A \cdot A'$).

54.4 Conclusion

We have identified several types of use for ontology matching results in solving heterogeneity problems. These are determined by the kind of operation to perform, e.g., merging ontologies, transforming data, and the context in which this performed, i.e., run time or design time. We have also replaced these operations within an alignment life cycle whose term is broader than one shot matching-processing. This has led to identify four distinct areas:

matching has been the subject of the work done in this work package so far and considered in other deliverable (D2.2.3);

evaluating has also be largely covered by this work package through deliverables D2.2.2, D2.2.4, and D2.2.9.

enhancing requires tools specifically designed to manipulate alignments; these tools are used mostly at design time and will be considered in Section 55;

storing and sharing requires tools to store and share alignments that can be used at design and run time; they will be presented in Section 61;

processing requires tools to process alignments that can be used at desing and run time; they will be presented in Section 56 through Section 60;

Chapter 55

Design time support for ontology matching

The first place where ontology heterogeneity can be found is while designing an application. Ontology management environments [Waterfeld *et al.*, 2008] must support users in obtaining alignments and manipulating them.

There exist infrastructures which use alignments as one of their components. The goal of such infrastructures is to enable users to perform high-level tasks which involve generating, manipulating, composing and applying alignments within the same environment. These infrastructures usually rely on the implementation of specific operations (merging, transforming, etc.) which will be considered in further chapters.

We consider here two types of infrastructures: frameworks which usually encompass matching and processing alignments (including sophisticated manipulations of these alignments) and editors which allow end users to control and modify alignments before processing them.

We illustrate the various frameworks supporting design time matching and alignment exploitation (§55.1) and alignment editors (§55.2).

55.1 Frameworks

Frameworks are environments providing support for alignment management. They usually provide formats and API so that developers can introduce matching algorithms and/or alignment processors in the framework.

55.1.1 Model management

Model management [Bernstein *et al.*, 2000; Madhavan *et al.*, 2002; Melnik, 2004] has been promoted in databases for dealing with data integration in a generic way. It offers a high-level view to the operations applied to databases and their relations. It aims at providing a metadata manipulation infrastructure in order to reduce the amount of programming required to build metadata driven applications.

Model management deals with *models* which can be related by *mappings*. A *model* is an information structure, such as an XML schema, a relational database schema, a UML model (by extrapolation, we will consider that it could be an ontology). Similarly, *mappings* are oriented alignments from one model into another. Technically, a key idea of generic model management is to solve metadata intensive tasks at a high level of abstraction using a concise script. It is generic in the sense that a single implementation should be applicable to the majority of data models and scenarios, e.g., data translation, data integration. However, it is primarily targeted at databases. It provides an algebra to manipulate models and mappings. In [Melnik *et al.*, 2005], the following operators are defined:

- $\mathsf{Match}(m, m')$ which returns the mapping a between models m and m';
- $\mathsf{Compose}(a, a')$ which composes mappings a and a' into a new one a'', given that the range of a' is the domain of a;
- $\mathsf{Confluence}(a, a')$ which merges alignments by union of non conflicting correspondences, provided as a and a' that have the same domain and range;
- $\mathsf{Merge}(a, m, m')$ which merges two models m and m' according to mapping a;
- $\mathsf{Extract}(a, m)$ which extracts the portion of model m which is involved in mapping a;
- $\mathsf{Diff}(a, m)$ which extracts the portion of model m which is not involved in mapping a.

A mapping in this context is a function from m to m'. [Melnik *et al.*, 2005] also provides axioms governing these operations. For instance, the merge operation between two models m' and m'' through a mapping a, returns a new model $m = \mathsf{Domain}(a') \cup \mathsf{Domain}(a'')$ and a pair of surjective mappings a' and a'' from m to m' and m'' respectively, such that: $a = \mathsf{Compose}(\mathsf{Invert}(a'), a'')$.

A typical example of the model management script is as follows:

```
A1 := Match( O1, O2 );
A2 := Match( O2, O3 );
O4 := Diff( O1, A1 );
A3 := Compose( A1, A2 );
O5 := Merge( Extract( O1, A1), O3, A3 );
O6 := Merge( O4, O5, ∅ );
```

The above example operates with three ontologies. It merges the first one and the last one on the basis of the composition of their alignment with the intermediate one. Finally, it adds the part of the first one that was not brought in the first merge.

There are some model management systems available. In particular, Rondo[1] is a programming platform implementing generic model management [Melnik *et al.*, 2003b; Melnik *et al.*, 2003a]. It is based on conceptual structures which constitute the main Rondo abstractions:

Models such as relational schemas, XML schemas, are internally represented as directed labelled graphs, where nodes denote model elements, e.g., relations and attributes. Each such element is identified by an object identifier (OID).

Morphisms are binary relations over two, possibly overlapping, sets of OIDs. The morphism is typically used to represent a mapping between different kinds of models. Morphisms can always be inverted and composed.

Selectors are sets of node identifiers from a single or multiple models. These are denoted as S. A selector can be viewed as a relation with a single attribute, $S(V : OID)$, where V is a unique key.

[1]http://infolab.stanford.edu/ modman/rondo/

The operators presented above, e.g., match, merge, are implemented upon these conceptual structures. Match is implemented in Rondo by using the Similarity flooding algorithm [Melnik *et al.*, 2002]. Rondo is currently a standalone program with no editing functions.

Another system, called Moda, is described in [Melnik *et al.*, 2005] in which correspondences are expressed as logical formulas. This system is more expressive than Rondo. Examples of some other model management systems include: GeRoMe [Kensche *et al.*, 2005] and ModelGen [Atzeni *et al.*, 2005; Atzeni *et al.*, 2006].

55.1.2 COMA++ (University of Leipzig)

COMA++ [Do and Rahm, 2002; Do, 2005] is another standalone (schema) matching workbench that allows integrating and composing matching algorithms. It supports matching, evaluating, editing, storing and processing alignments.

It is built on top of COMA [Do and Rahm, 2002] and provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers. COMA++ enables importing, storing and editing schemas (or models). It also allows various operations on the alignments among which compose, merge and compare. Finally, alignments can be applied to schemas for transforming or merging them.

Contrary to Rondo, the matching operation is not described as atomic but rather described as workflow that can be graphically edited and processed. Users can control the execution of the workflow in a stepwise manner and dynamically change execution parameters. The possibility of performing iterations in the matching process assumes interaction with users who approve obtained matches and mismatches to gradually refine and improve the accuracy of match (see Figure 55.1). The matching operation is performed by the Execution engine based on the settings provided by the Match customiser, including matchers to be used and match strategies.

The data structures are defined in a homogeneous proprietary format. The Schema pool provides various functions to import and export schemas and ontologies and save them to and from the internal Repository. Similarly, the Mapping pool provides functions to manipulate mappings. COMA++ can also export and import the matching workflows as executable scripts (similar to those manipulated in Rondo).

Finally, according to [Do, 2005], there are some other tools built on top of COMA++. For example, the CMC system provides a new weighting strategy to automatically combine multiple matchers [Tu and Yu, 2005], while the work of [Dragut and Lawrence, 2004] has adapted COMA to compute correspondences between schemas by composing the correspondences between individual schemas and a reference ontology.

55.1.3 MAFRA (Instituto Politecnico do Porto and University of Karlsruhe)

MAFRA[2] (MApping FRAmework) is an interactive, incremental and dynamic framework for matching distributed ontologies [da Silva, 2004; Mädche *et al.*, 2002]. It proposes an architecture for dealing with "semantic bridges" that offers functions such as creating, manipulating, storing and processing such bridges. MAFRA does not record alignments in a non processable format but associates transformations with bridges. MAFRA does not offer editing or sharing alignments.

[2]http://mafra-toolkit.sourceforge.net

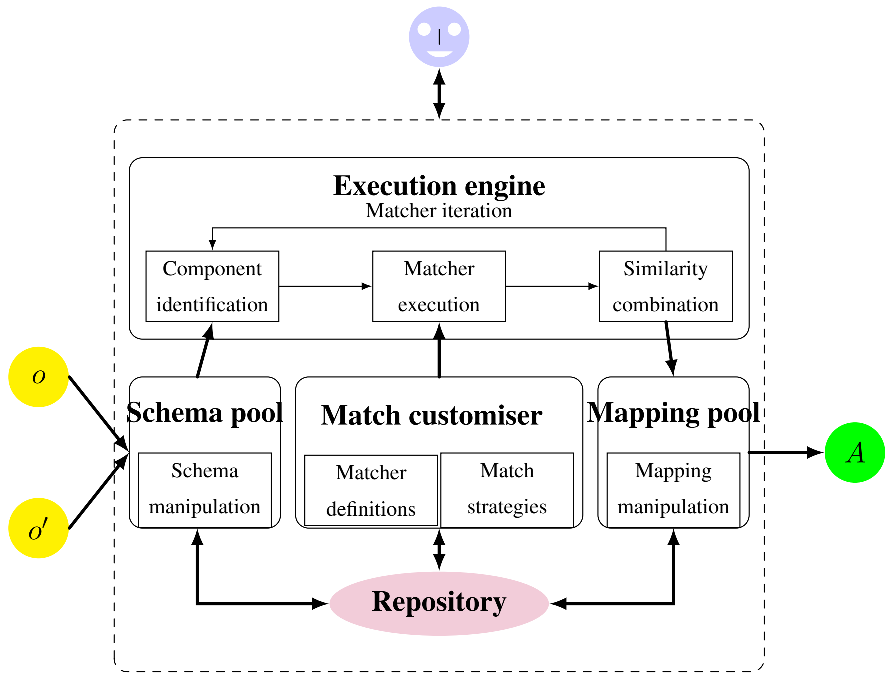


Figure 55.1: COMA++ architecture (adapted from [Do, 2005]).

The framework consists of horizontal and vertical dimensions. The horizontal dimension covers the mapping process. It is organised according to the following components:

– *Lift and Normalisation.* This module handles syntactic, structural, and language heterogeneity. In particular, the lifting process includes translation of input ontologies into an internal knowledge representation formalism, which is RDF Schema. Normalisation, in turn, includes (i) tokenisation of entities, (ii) elimination of stop words, (iii) expansion of acronyms.

– *Similarity.* This module calculates similarities between ontology entities by exploiting a combination of multiple matchers. First, *lexical similarity* between each entity from the source ontology and all entities from the target ontology is determined based on WordNet and altered Resnik measure. Second, the *property similarity* is computed. This measures similarity between concepts based on how similar the properties they are involved in are. Finally, *bottom-up* and *top-down similarities* are computed. For example, bottom-up matchers take as input the property (dis)similarity and propagate it from lower parts of the ontology to the upper concepts, thus yielding an overall view of similarity between ontologies.

– *Semantic Bridging.* Based on the similarities determined previously, the correspondences (bridges) between the entities of the source and target ontologies are established. Bridges, in turn, can be executed for the data translation task.

– *Execution.* The actual processing of bridges is performed in the execution module. This module translates instances from the source ontology to the target ontology. This translation can either be performed off-line, i.e., one time transformation, or on-line, i.e., dynamically, thus taking into account the 'fresh' data, if any.

– *Post-processing.* This module is in charge of the analysis and improvement of the transformation results, for instance, by recognising that two instances represent the same real-world object.

Components of the vertical dimension interact with horizontal modules during the whole mapping process. There are four vertical components. The *Evolution* module, in a user-assisted way, synchronises bridges obtained with the Semantic Bridging module according to the changes in the source and target ontologies. The *Cooperative Consensus Building* module helps users to select the correct mappings, when multiple mapping alternatives exist. The *Domain Constraints and Background Knowledge* module stores common and domain specific knowledge, e.g., WordNet, precompiled domain thesauri, which are used to facilitate the similarity computation. Finally, a graphical user interface assists users in accomplishing the matching process with a desired quality.

55.1.4 Alignment API (INRIA Rhône-Alpes)

The Alignment API and implementation[3] [Euzenat, 2004] offer matching ontologies, manipulating, storing and sharing alignments as well as processor generation. It also features an Alignment Server (see Section 61.3) which can be accessed by clients through API, web services, agent communication languages ot HTTP. It does not support editing alignments.

The Alignment API manipulates alignments in the Alignment format. It can be used for can be used for implementing this format and linking to alignment algorithms and evaluation procedures. It defines a set of interfaces and a set of functions that they can perform.

Classes

The OWL API is extended with the org.semanticweb.owl.align package which describes the Alignment API. This package name is used for historical reasons. In fact, the API itself is fully independent from OWL or the OWL API.

The Alignment API is essentially made of three interfaces:

Alignment describes a particular alignment. It contains a specification of the alignment and a list of cells.

Cell describes a particular correspondence between entities.

Relation does not mandate any particular feature.

To these interfaces implementing the Alignment format, are added several of other interfaces:

AlignmentProcess extends the Alignment interface by providing an align method. So this interface is used for implementing matching algorithms (Alignment can be used for representing and manipulating alignments independently of algorithms).

Evaluator describes the comparison of two alignments (the first one could serve as a reference). Each implemented measure must provide the eval method.

An additional AlignmentException class specifies the kind of exceptions that are raised by alignment algorithms and can be used by alignment implementations.

[3]http://alignapi.gforge.inria.fr

Functions

The Alignment API provides support for manipulating alignments. As in [Bechhofer *et al.*, 2003], these functions are separated in their implementation. It offers the following functions:

Parsing and serialising an alignment from a file in RDF/XML (AlignmentParser.read(), Alignment.write());

Computing the alignment, with input alignment (Alignment.align(Alignment, Parameters));

Thresholding an alignment with threshold as argument (Alignment.cut(double));

Hardening an alignment by considering that all correspondences whose strength is strictly greater than the argument are converted to $\top$, while the others are converted to $\bot$ (Alignment.harden(double));

Comparing one alignment with another (Evaluator.eval(Parameters)) and serialising the result (Evaluator.write());

Outputting alignments in a particular format, e.g., SWRL, OWL, XSLT, RDF. (Alignment.render(visitor));

Matching and evaluation algorithms accept parameters. These are put in a structure that allows storing and retrieving them. The parameters can be various weights used by some algorithms, some intermediate thresholds or the tolerance of some iterative algorithms. There is no restriction on the kind of parameters to be used.

The Alignment API has been implemented in Java. This implementation has been used for various purposes: on-line alignment [Zhdanova and Shvaiko, 2006] and Evaluation tool in the Ontology Alignment Evaluation Initiative [Euzenat *et al.*, 2004b; Stuckenschmidt *et al.*, 2005; Shvaiko *et al.*, 2007]. Also many extensions use it for implementing matching algorithms, such as oMap [Straccia and Troncy, 2005b], FOAM [Ehrig, 2007], and OLA [Euzenat and Valtchev, 2004].

55.1.5 FOAM (University of Karlsruhe)

FOAM[4] [Ehrig *et al.*, 2005; Ehrig, 2007] is a framework in which matching algorithms can be integrated. It mostly offers matching and processor generation. It does not offer on-line services nor alignment editing, but is available as a Prompt plug-in (see Section 55.2.4) and is integrated in the KAON2 ontology management environment.

FOAM is a general tool for processing similarity-based ontology matching. It follows a general process, presented in Figure 55.2, which is made of the following steps:

Feature engineering selects the features of the ontologies that will be used for comparing the entities.

Search step selection selects the pairs of elements from both ontologies that will be compared.

Similarity computation computes the similarity between the selected pairs using the selected features.

Similarity aggregation combines the similarities obtained as the result of the previous step for each pair of entities.

[4]http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/

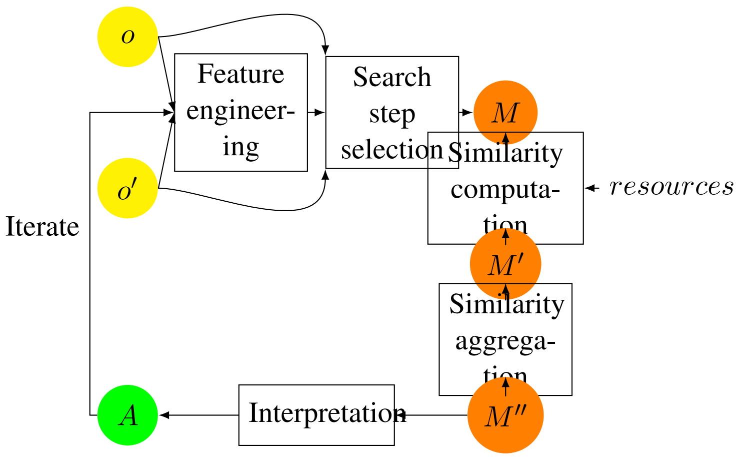


Figure 55.2: FOAM architecture (adapted from [Ehrig, 2007]).

Interpretation extracts an alignment from the computed similarity.

Iteration iterates this process, if necessary, taking advantage of the current computation.

The FOAM framework bundles several algorithms and strategies developed by its authors. Within this framework have been cast matching systems such as NOM [Ehrig and Sure, 2004], QOM [Ehrig and Staab, 2004], and APFEL [Ehrig *et al.*, 2005]. More systems can be integrated simply by changing any of the modules above. The global behaviour of the system can be parameterised through different scenarios, e.g., data integration, ontology merging, ontology evolution, query rewriting and reasoning. FOAM offer default parameters adapted to these tasks.

FOAM itself is based on the KAON2 [Oberle *et al.*, 2004] suite of tools and accepts ontologies in the OWL-DLP fragment. It offers a web-based interface. Finally, it also offers translation tools from and to the Alignment format [Euzenat, 2004] and other formats.

Platforms for integrating matchers and alignment manipulation operations are relatively new, however, they constitute a promising perspective to knowledge engineers and application developers. Another, type of useful alignment manipulation systems are alignment editors which offer human users the opportunity to be involved in the matching process.

55.2 Ontology editors with alignment manipulation capabilities

Other tools for dealing with ontology matching are ontology edition environments provided with support for matching and importing ontologies. These tools are primarily made for creating ontologies, but they also provide tools for comparing ontologies and relating them.

55.2.1 Web Service Modeling Toolkit (DERI, Austria)

WSMT[5] is a design time alignment creator and editor. It manipulates the AML [Scharffe and de Bruijn, 2005] format and can generate WSML rules [de Bruijn, 2007]. It also works as a

[5]http://wsmt.sourceforge.net

standalone system.

As mentioned above, data mediation within a semantic environment such as WSMX is a semi-automatic process where alignments between two ontologies are created at design time and then applied at run time in order to perform instance transformation in an automatic manner. Approaches for automatic generation of ontology alignments do exist but their accuracy is usually unsatisfactory for business scenarios and it is necessary for business to business integration to have an engineer involved in creating and validating the correspondences between ontologies. This is a non-trivial task and the user should be guided through the process of creating these alignments and ensuring their correctness.

Web Service Modeling Toolkit (WSMT) [Kerrigan *et al.*, 2007] is a semantic web service and ontology engineering toolkit, also featuring tools capable of producing alignments between ontologies based on human user inputs. It offers a set of methods and techniques that assist domain experts in their work such as different graphical perspectives over the ontologies, suggestions of the most related entities from the source and target ontology, guidance throughout the matching process [Mocan *et al.*, 2006]. The tools and the domain expert work together in an iterative process that involves cycles consisting of suggestions from the tool side and validation and creation of correspondences from the domain expert side.

Within WSMT, alignments are expressed by using the Abstract Mapping Language (AML) [Scharffe and de Bruijn, 2005] which is a formalism-neutral syntax for ontology alignments. WSMT includes several tools and editors meant to offer all the necessary support for editing and managing such ontology alignments:

Alignment validation: WSMT provides validation for the AML syntax useful especially when alignments created in various tools need to be integrated into the same application.

Alignment text editor: This editor provides a text editor for the human readable syntax of AML. It provides similar features to that of a programming language editor, e.g., a Java editor, including syntax highlighting, in line error notification, content folding and bracket highlighting. This editor enables the engineer to create or modify correspondences through textual descriptions. Such a tool is normally addressed to experts familiar with both the domain and the alignment language.

Alignment view-based editor: The View-based Editor provides graphical means to create correspondences between ontologies. Such a tool is addressed to those experts that are capable of understanding the problem domain and who can successfully align the two heterogeneous ontologies but they are not specialists in logical languages as well. Additionally, even if domain experts have the necessary skills to complete the alignment by using a text editor, a graphical mapping tool would allow them to better concentrate on the heterogeneity problems to be solved and, in principle, to maximise the efficiency of the overall mapping process. All the advantages described above, have been acknowledged by other approaches as well [Mädche *et al.*, 2002; Noy and Musen, 2003]. The View-based Editor includes some of well-established classical methods, e.g., lexical and structural suggestion algorithms, iterative alignment creation processes. Additionally, this particular approach provides several new concepts and strategies aiming to enhance the overall automation degree of the ontology matching tool [Mocan and Ciampian, 2005]. Three of the most important features of this tool (views, decomposition and contexts) are presented below.

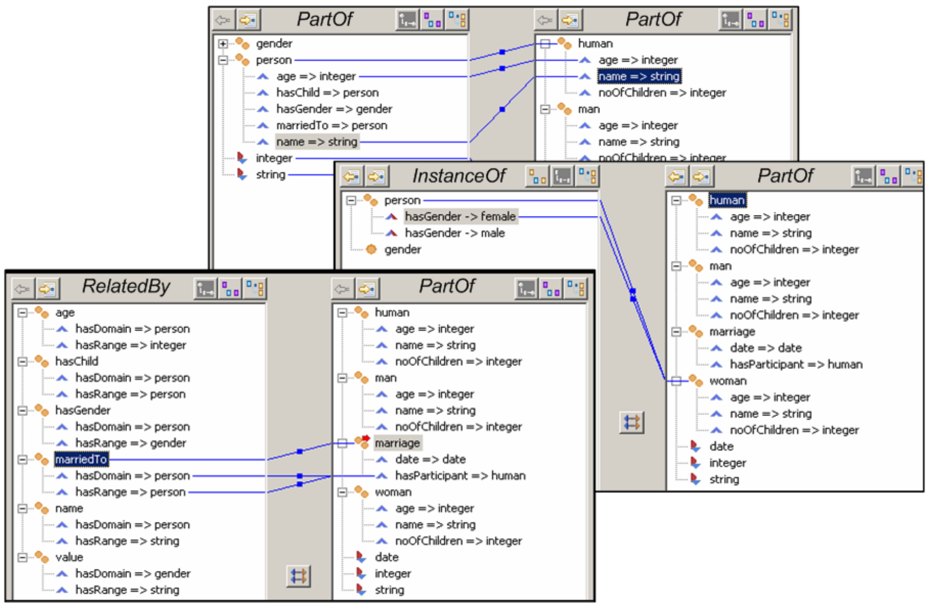


Figure 55.3: Mapping views in the AML View-Based Editor.

A *view* (also referred to as a perspective in [Mocan *et al.*, 2006]) represents a viewpoint in displaying the entities defined in a particular ontology; each view displays entities from the ontology in a two-level tree structure. The graphical viewpoint adopted to visualise the source and the target ontologies is important to simplify the design of the correspondences according to their type. By switching between combinations of these views on the source and the target ontologies, certain types of correspondences can be created using the same operations, combined with mechanisms for ontology traversal and contextualised visualisation strategies.

Each view specifies what ontological entities should appear as roots or as children in these trees, by switching the focus between various relationships existing in the ontology. Views can be defined and grouped in pairs in such a way to solicit specific skill sets, offering support for users profiling. Currently, three types of views are available, namely PartOf (concepts as roots and their attributes as children), InstanceOf (concepts as roots and their attributes together with the values they can take as children) and RelatedBy (attributes as roots and their domain or range as children); Figure 55.3 illustrates the creation of alignments by using combinations of these perspectives.

Decomposition is the process of bringing into focus the descriptive information of the root items presented in the view tree by exploring their children. A successful decomposition is followed by a context update. That is, instead of displaying the whole ontology at a time, only a subset (the one determined by decomposition) can be presented. Such subsets form the source and target contexts. If views can be seen as a vertical projection over ontologies, contexts can be seen as a horizontal projection over views. Decomposition and contexts aims to improve the effectiveness of the matching process by keeping the domain expert focused on the exact heterogeneity problem to be solved and by ensuring that all the problem-related entities have been explored.

Mappings Views: The Mappings Views provide a light overview on the alignment created either by using the Text Editor or the View-based Editor. Instead of seeing the full description of an alignment (as quadruples in AML syntax or grounded rules in an ontology language) the domain expert can choose to see a more condensed version of this information: which are the entities in the source and in the target that are matched and if there are some special conditions associated with them.

Once a satisfying alignment has been designed, it can be stored and managed so that it is available to whoever needs it.

55.2.2 Chimaera (Stanford University)

Chimaera is a browser-based environment for editing, merging and testing (diagnosing) large ontologies [McGuinness *et al.*, 2000]. It aims to be a standard-based and generic tool. Users are provided with a graphical user interface (the Ontolingua ontology editor) for editing taxonomy and properties. They also can use various diagnosis commands, which provide a systematic support for pervasive tests and changes, e.g., tests for redundant super classes, slot value or type mismatch. Matching in the system is performed as one of the subtasks of a merge operation. Chimaera searches for merging candidates as pairs of matching terms, with terminological resources such as term names, term definitions, possible acronym and expanded forms, names that appear as suffixes of other names. It generates name resolution lists that help users in the merging task by suggesting terms which are candidates to be merged or to have taxonomic relationships not yet included in the merged ontology. The suggested candidates can be names of classes or slots. The result is output in OWL descriptions. Chimaera also suggests taxonomy areas that are candidates for reorganisation. These edit points are identified by using heuristics, e.g., looking for classes that have direct subclasses from more than one ontology.

55.2.3 The Protégé Prompt Suite (Stanford University)

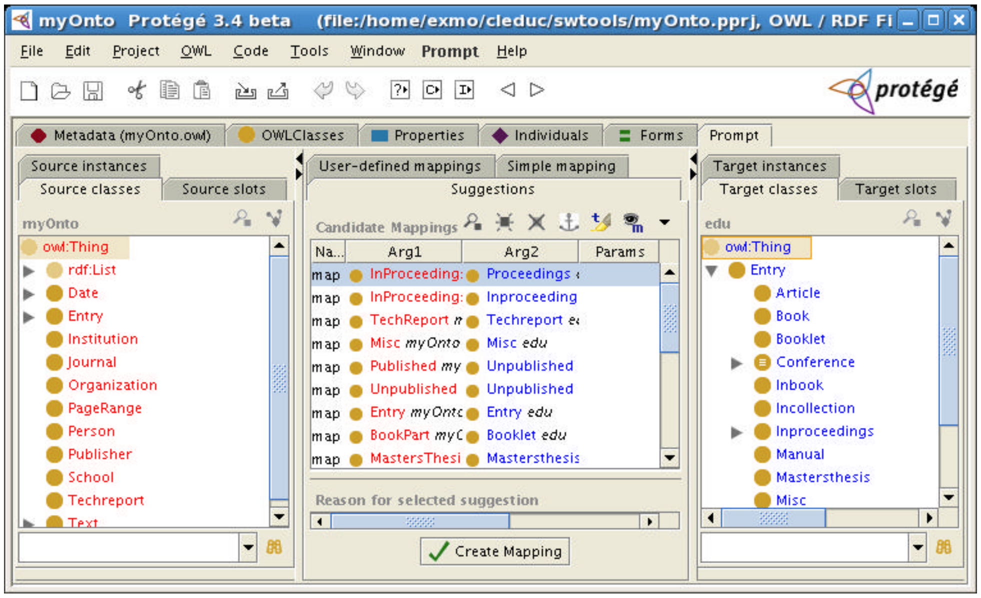
Protégé[6] is an ontology edition environment that offers design time support for matching. In particular it features Prompt[7] [Noy and Musen, 2003], an interactive framework for comparing, matching, merging, maintaining versions, and translating between different knowledge representation formalisms [Noy and Musen, 2003; Noy, 2004]. The Prompt suite includes: an alignment editor (see Figure 55.4), an interactive ontology merging tool, called iPrompt [Noy and Musen, 2000] (formerly known as Prompt), an ontology matching tool, called Anchor-Prompt [Noy and Musen, 2001], an ontology-versioning tool, called PromptDiff [Noy and Musen, 2002b], and a tool for factoring out semantically complete subontologies, called PromptFactor.

Since alignments are expressed in an ontology, they can be stored and shared through the Protégé server mode. Similarly to Protégé, Prompt can be extended through plug-ins. For example, there is a Prompt plug-in for FOAM. As a recent extension, Prompt offers a new functionality which allows to edit alignments graphically. The alignments can be considered as suggestions from which users may select appropriate correspondences. The graphical alignment editor is named CogZ [Falconer and Storey, 2007]. CogZ allows to create alignments by hand through drag-and-drop, to visualise the alignments and selectively filter some of the correspondences.

Figure 55.5 shows an example of usage of the graphical mapping editor in Prompt.

[6]http://protege.stanford.edu/

[7]http://protege.stanford.edu/plugins/prompt/prompt.html


myOnto Protégé 3.4 beta (file:/home/exmo/cleduc/swtools/myOnto.pprj, OWL / RDF Fi
File Edit Project OWL Code Tools Window Prompt Help
protégé
Metadata (myOnto.owl) OWLClasses Properties Individuals Forms Prompt
Source instances
Source classes Source slots
User-defined mappings Simple mapping
Suggestions
Target instances
Target classes Target slots
myOnto
owl:Thing
rdf:List
Date
Entry
Institution
Journal
Organization
PageRange
Person
Publisher
School
Techreport
Text
Candidate Mappings
Reason for selected suggestion
Create Mapping
edu
owl:Thing
Entry
Article
Book
Booklet
Conference
Inbook
Incollection
Inproceedings
Manual
Mastersthesis
Misc

Na...	Arg1	Arg2	Params
map	InProceeding:	Proceedings (	
map	InProceeding:	Inproceeding	
map	TechReport n	Techreport ee	
map	Misc myOnto	Misc edu	
map	Published my	Unpublished	
map	Unpublished	Unpublished	
map	Entry myOntc	Entry edu	
map	BookPart myC	Booklet edu	
map	MastersThesi	Mastersthesis	

Figure 55.4: Prompt alignment editor: represents and synchronises two ontologies through their sets of correspondences.

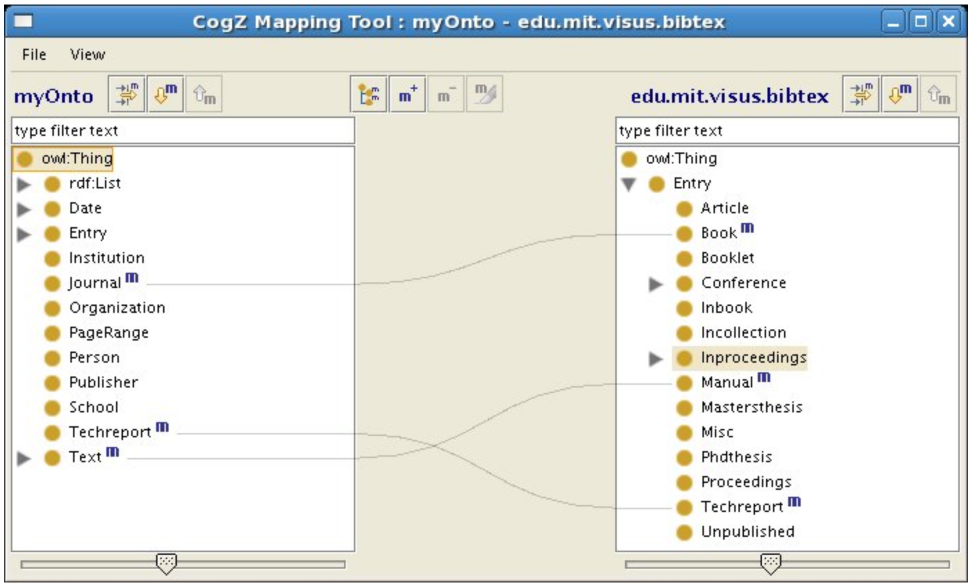


Figure 55.5: Protégé CogZ alignment editor: curved lines connecting entities within two ontologies are correspondences that are created by providing subsumption relationships between the entities.

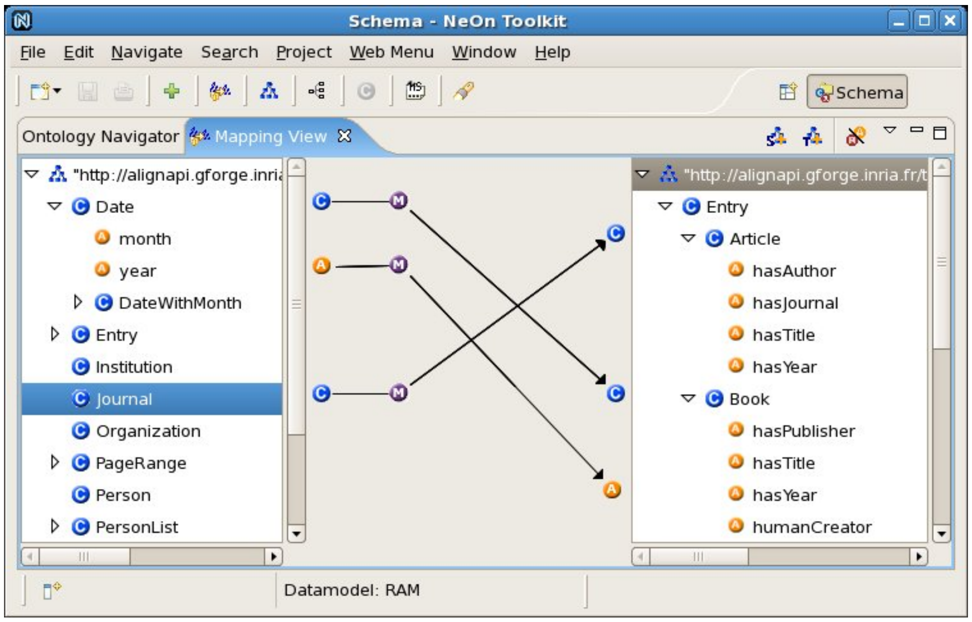


Figure 55.6: NeOn toolkit OntoMap editor: it allows to manipulate alignments between concepts and attributes and to visualise them.

55.2.4 The NeOn toolkit (NeOn Consortium)

The NeOn toolkit is an environment for managing networked ontologies developed within the NeOn project[8]. It is developed as a plug-in for managing ontologies under Eclipse and extends previous products such as KAON2 [Oberle *et al.*, 2004] which is the basis for a tool called OntoStudio [Waterfeld *et al.*, 2008].

The NeOn toolkit features run time and design time ontology alignment support. It can be extended through a plug-in mechanism, so it can be customized to the user's needs. As a development environment for ontology management, the NeOn Toolkit supports the W3C recommendations OWL and RDF as well as F-Logic for processing rules. With the support of the integrated mapping-tool, named OntoMap, heterogeneous data sources, e.g., databases, file systems, UML diagrams, can be connected to ontologies quickly and easily. Thus it provides a single view through the ontology to all connected data sources. For example, the Neon Toolkit can import external database schemes and convert them into ontologies. Connected by rules, relations of content from different databases can be created.

In addition, the Neon Toolkit offers a graphical editor for alignments, called OntoMap, in which the user can create, delete and store alignments. OntoMap allows for the creation of different types of alignments: from concept to concept (CC), from attribute to attribute (AA), etc. Each alignment can be edited by defining a filter that gives the user the possibility to limit the matched instances over their characteristic values. Figure 55.6 shows the OntoMap editor that has created CC and AA mappings.

[8]http://www.neon-project.org

55.3 Conclusion

This chapter has presented tools for dealing with ontology alignment at design time. These environments are controled by a human person and rely for their implementation on various operations. We have detailed operations on alignments such as editing or trimming. These frameworks also offers operations which process alignments on actual ontologies or databases. Such operations involve:

- merging ontologies (§56);
- transforming ontologies (§57);
- translating data and instances (§58);
- mediating queries and answers (§59);
- reasoning on aligned ontologies (§60).

These operations can either be applied at run time or at design time. This is the reason why they have not been presented in detail and will be presented in individual chapters. They will be completed by the functions of storing and sharing ontologies (§61) which are not alignment processing per se, but are useful at both run time and design time.

Chapter 56

Ontology merging

There are cases in which the ontologies are not kept separate but need to be merged into a single new ontology. As an example, we can consider the case of one vendor acquiring another, their catalogs will probably be merged into a single one. Ontology merging is achieved by taking the two ontologies to be merged and an alignment between these two ontologies. It results in a new ontology combining the two source ontologies.

56.1 Specification

Ontology merging is a first natural use of ontology matching. As depicted in Figure 56.1, it consists of obtaining a new ontology o'' from two matched ontologies o and o' so that the matched entities in o and o' are related as prescribed by the alignment. Merging can be presented as the following operator:

$$\mathsf{Merge}(o, o', A) = o''$$

The ideal property of a merge would be that ($\models$ being the consequence relation such as defined in [Euzenat and Shvaiko, 2007] or [Bouquet *et al.*, 2004a]):

$$\begin{aligned}
&\mathsf{Merge}(o, o', A) \models o \\
&\mathsf{Merge}(o, o', A) \models o' \\
&\mathsf{Merge}(o, o', A) \models \alpha(A)
\end{aligned}$$

if $\alpha(A)$ is the alignment expressed in the logical language of $\mathsf{Merge}(o, o', A)$, and

$$o, o', A \models \mathsf{Merge}(o, o', A)$$

The former set of assertions means that the merge preserves the consequences of both ontologies and of the relations expressed by the alignment. The latter assertion means that the merge does not entail more consequences than specified by the semantics of alignments [Zimmermann and Euzenat, 2006]. Of course, this is not restricted to the union of the consequences of o, o' and A.

When the ontologies are expressed in the same language, merging often involves putting the ontologies together and generating bridge or articulation axioms.

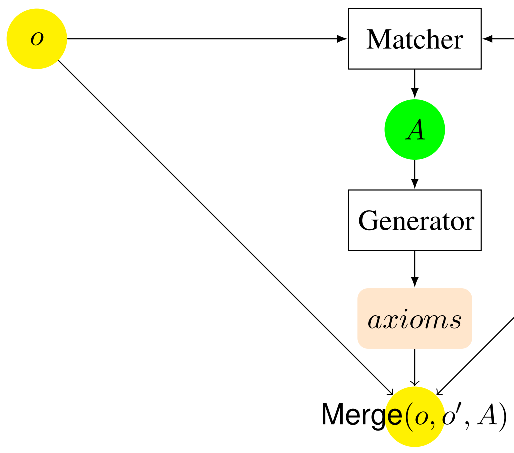


Figure 56.1: Ontology merging (from [Euzenat and Shvaiko, 2007]). From two matched ontologies o and o', resulting in alignment A, articulation $axioms$ are generated. This allows the creation of a new ontology covering the matched ontologies.

Merging does not usually require a total alignment: those entities which have no corresponding entity in the other ontology will remain unchanged in the merged ontology.

The ontology merging process can be fully automatic if an adequate alignment is provided [Scharffe, 2007], but usually requires human intervention in order to solve conflicts and choose a merging strategy. Figure 56.1 illustrates the ontology merging process.

Ontology merging is especially used when it is necessary to carry out reasoning involving several ontologies. It is also used when editing ontologies in order to create ontologies tailored for a particular application. In such a case, it is most of the time followed by a phase of ontology reengineering, e.g., suppressing unwanted parts from the obtained ontology.

56.2 Systems

Protégé [Noy and Musen, 2003; Noy, 2004] and Rondo [Melnik *et al.*, 2002] offer independent operators for ontology merging. The Alignment API [Euzenat, 2004] can generate axioms in OWL or SWRL for merging ontologies.

OntoMerge [Dou *et al.*, 2005] is a system fully dedicated to merging ontologies. Merging two ontologies is performed by taking the union of the axioms defining them. Bridge axioms or bridge rules are then added to relate the terms in one ontology to the terms in the other. They can be expressed using the full power of predicate calculus. It is assumed that bridge rules are provided by domain experts, or by other matching algorithms, which are able to discover and interpret them with clear semantics (OntoMerge does not offer matching).

Other systems are able to match ontologies and merge them directly: FCA-merge [Stumme and Mädche, 2001], SKAT [Mitra *et al.*, 1999], DIKE [Palopoli *et al.*, 2003], HCONE [Kotis *et al.*, 2006]. OntoBuilder [Modica *et al.*, 2001] uses ontology merging as an internal operation: the system creates an ontology that is mapped to query forms. This ontology is merged with the global ontology so that queries can be directly answered from the global ontology.

Chapter 57

Ontology transformation

Ontology transformation is used for connecting an ontology to another ontology. It computes the difference between the ontologies for obtaining an ontology corresponding to their merge but preserving only the specific part of one of the two ontologies (it is thus usually smaller than the merge).

57.1 Specification

Ontology transformation, from an alignment A between two ontologies o and o', consists of generating an ontology o'' expressing the entities of o with respect to those of o' according to the correspondences in A. It can be denoted as the following operator:

$$\mathsf{Transform}(o, A) = o''$$

Contrary to merging, ontology transformation, and the operators to follow, are oriented. This means that the operation has an identified source and target and from an alignment it is possible to generate two different operations depending on source and target.

The result is rather an ontology featuring only the elements of the source ontology which are not equivalent (according to the alignment) to an element of the target ontology. What is expected is that:

$$\mathsf{Merge}(o, o', A) \equiv \mathsf{Transform}(o, A) \cup o'$$

57.2 Example

Consider an ontology o' defining concepts e and e' as well as property a. Consider ontology o defining concepts c, c', c'' and c''' as well as property b and individual i and i'. The ontology o also contains the following axioms:

$$\begin{aligned} c'' &= (\text{and } c\ c'\ (\text{all } b\ 3)) \\ i &\in c'' \\ c''' &\leq c'' \\ i' &\in c''' \end{aligned}$$

Let us consider the alignment defined by:

$$\begin{aligned} e &= c \\ e' &= c' \\ a &= b \end{aligned}$$

Then the transformation of o with regard to this alignment is:

$$\begin{aligned} i &\in e \\ i &\in e' \\ i &\in (\text{all } a\ 3) \\ c''' &\leq (\text{and } e\ e'\ (\text{all } a\ 3)) \\ i' &\in c''' \end{aligned}$$

As explained before, the transformation has replaced elements of o by equivalent elements of o'. So the resulting ontology is expressed with regard to o'. This allows answering directly queries expressed with regard to o'.

57.3 Systems

Ontology transformation is not well supported by tools. It is useful when one wants to express one ontology with regard to another one. This can be particularly useful for connecting an ontology to a common upper level ontology, for instance, or local schemas to a global schema in data integration. This is rather a design time operation.

Chapter 58

Data translation

A very common operation after matching is data translation that allows to export data to another ontology. In fact data translation occurs in ontology merging (partially) and in query mediation.

58.1 Specification

Data translation, presented in Figure 58.1, consists of translating instances from entities of ontology o into instances of connected entities of matched ontology o'. This can be expressed by the following operator:

$$\textsf{Translate}(d, A) = d'$$

Data translation usually involves generating some transformation program from the alignment.

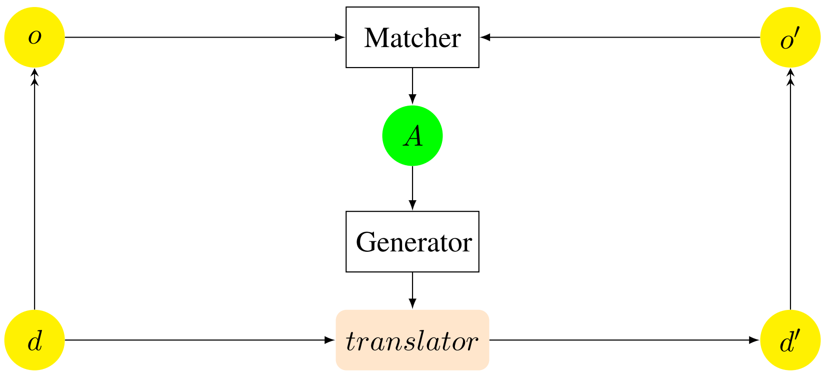


Figure 58.1: Data translation. From two matched ontologies o and o', resulting in alignment A, a $translator$ is generated. This allows the translation of the instance data (d) of the first ontology into instance data (d') for the second one.

Data translation requires a total alignment if one wants to translate all the extensional information of the source ontology. Non total alignments risk loosing instance information in the translation (this can also be acceptable if one does not want to import all the instance information).

Data translation is used for importing data under another ontology without importing the ontology itself. This is typically what is performed by database views in data integration, in multiagent

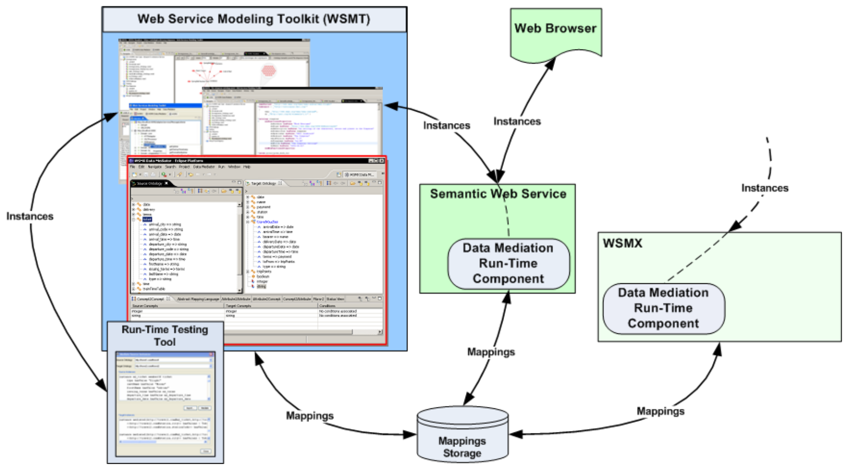


Figure 58.2: Run time data mediator usage scenario (from [Mocan and Cimpian, 2007]).

communication for translating messages, and in semantic web services for translating the flow of information in data mediators.

58.2 Example of instance translation

The mediation of the heterogeneous semantic data can be achieved through instance translation. Data represented by ontology instances has to be translated either by the sender or transparently by a third party in the format required by the receiver, i.e., instances expressed in the target ontology. In order to accommodate such a scenario, the alignments generated by using any ontology matching technique have to be processed by an engine able to perform instance translation. If the alignments are expressed in an abstract form, e.g., using AML or the Alignment format, an extra step has to be performed: the correspondences in the alignment must be expressed in a concrete ontology specification language which can be interpreted.

Figure 58.2 shows how such an instance translation engine (the Data Mediation Run Time Component in WSMX) can be deployed and used. A straightforward way is to integrate it in an information system (in this case WSMX) which needs translation support in order to facilitate the exchange of heterogeneous data. Another possibility is to encapsulate this engine in a (semantic) web service and to allow external calls having as inputs the source instances and optionally the alignments to be applied. As output, the corresponding target instances are returned. Additionally, such an engine can be used for testing the correctness of the alignments been produced, either by using it as a test module in the design time matching tool (see the WSMT MUnit) or by providing a web interface that would allow domain experts to remotely send source instances to be translated in target instances.

58.3 Instance identification

Data translation results in various sets of instances described according to the same ontology. The different origin of these instances may lead to duplicates. For instance, in a web application integrating various on-line catalogs, each described as an ontology, once the catalogs queried and the results adapted to the reference ontology, it is likely that some products are sold by many vendors. Similar products have to be identified in order to be presented under the same one (eventually with the different prices kept separate).

Instance unification techniques are used to merge similar instances by analyzing their attributes values, as well as the relations they share with other instances. This method is usable when one knows that the instances are the same. This works, for example, when integrating, two human resource databases of the same company, but does not apply for those of different companies or for databases of events which have no relations. A first natural technique for identifying instances is to take advantage of keys in databases. Keys can be either internal to the database, i.e., generated unique surrogates, in which case they are not very useful for identification, or external identification, in which case there is high probability that these identification keys are present in both data sets (even if they are not present as keys). In such a case, if they are used as keys, we can be sure that they uniquely identify an individual (like isbn). When keys are not available, or they are different, other approaches to determine property correspondences use instance data to compare property values. In databases, this technique has been known as record linkage [Fellegi and Sunter, 1969; Elfeky *et al.*, 2002] or object identification [Lim *et al.*, 1993]. They aim at identifying multiple representations of the same object within a set of objects. They are usually based on string-based and internal structure-based techniques used in ontology matching (see [Euzenat and Shvaiko, 2007]). If values are not precisely the same but their distributions can be compared, it is possible to apply global statistical techniques (see [Euzenat and Shvaiko, 2007]).

Instance identification is also necessary after two ontologies have been merged into one (see Chapter 56). Instances of the source ontologies then also need to be merged, and duplicates removed.

58.4 Systems

Rondo (§55.1.1) provides tools for data translation. The Alignment API (§55.1.4) can generate translations in XSLT or C-OWL. Many tools developed for data integration can generate translators under the form of SQL queries. Drago [Serafini and Tamilin, 2005] is an implementation of C-OWL, which can process alignments expressed in C-OWL for transferring data from one ontology to another one.

Some tools provide their output as data translation or process themselves the translation. These include: Clio [Miller *et al.*, 2000], ToMAS [Velegrakis *et al.*, 2003], TransScm [Milo and Zohar, 1998], MapOnto [An *et al.*, 2006], COMA [Do and Rahm, 2002], SKAT [Mitra *et al.*, 1999], and sPLMap [Nottelmann and Straccia, 2005].

Meanwhile, most of the commercially available ontology integration tools focus on automation of alignment processing, by opposition to matching. They are very often specialised in a particular segment of the matching space. Altova MapForce[1] and Stylus Studio XSLT Mapper[2] are specialised in XML integration. They integrate data from XML sources as well as databases or other structured sources. Microsoft BizTalk Schema Mapper[3] is targeted at the business process and information integration, using the proprietary BizTalk language. Ontoprise SemanticIntegrator[4] offers ontology-based integration of data coming from databases or ontologies. There are unfortunately no scholar references describing these systems in depth and URLs change so often that we refer the reader to www.ontologymatching.org for accurate and up to date information.

The matching operation itself is not automated within these tools, though they facilitate manual matching by visualising input ontologies (XML, database, flat files formats, etc.) and the correspondences between them. Once the correspondences have been established it is possible to specify, for instance, data translation operations over the correspondences such as adding, multiplying, and dividing the values of fields in the source document and storing the result in a field in the target document.

[1]http://www.altova.com/products/mapforce/data_mapping.html
[2]http://www.stylusstudio.com/xslt_mapper.html
[3]http://www.microsoft.com/biztalk/
[4]http://ontoedit.com

Chapter 59

Mediation

In this chapter, we consider a mediator as an independent software component that is introduced between two other components in order to help them interoperate.

59.1 Specification

There are many different forms of mediators, including some acting as brokers or dispatchers. We concentrate here on query mediators. Query mediation consists of rewriting a query q in terms of a source ontology o into terms of a target ontology o' (according to some alignment A). This corresponds to performing the following operation (Figure 59.1 illustrates this process):

$$\mathsf{TransformQuery}(q, A) = q'$$

TransformQuery is a kind of ontology transformation (see Chapter 57) which transforms a query expressed using ontology o into a query expressed with the corresponding entities of a matched ontology o'. Query rewriting has been largely studied in database integration [Duschka and Genesereth, 1997].

Once the rewritten query addressed to the target ontology, the instances eventually returned are described in terms of o'. They might have to be translated to instances of o in order to be further processed by the system. This consists of applying a data translation operation as described in Chapter 58:

$$\mathsf{Translate}(a', \mathsf{Invert}(A)) = a$$

Instance translation is done by taking instances described under a source ontology o', and translating them to instances of a target ontology o using the alignment between the two ontologies. In this case, the alignment is taken in the reverse direction as in the query transformation operation. New instances of o classes are described, and attribute values are transformed [Scharffe and de Bruijn, 2005] according to the alignment. This process may lead to the creation of multiple target instances for one source instance, or, inversely, to combine some source instances into one target instance. Instance transformation, illustrated in Figure 59.1, is used in the example scenario in Section 54.2.

The Translate operation performs data translation on the answer of the query if necessary. This process is presented in Figure 59.1.

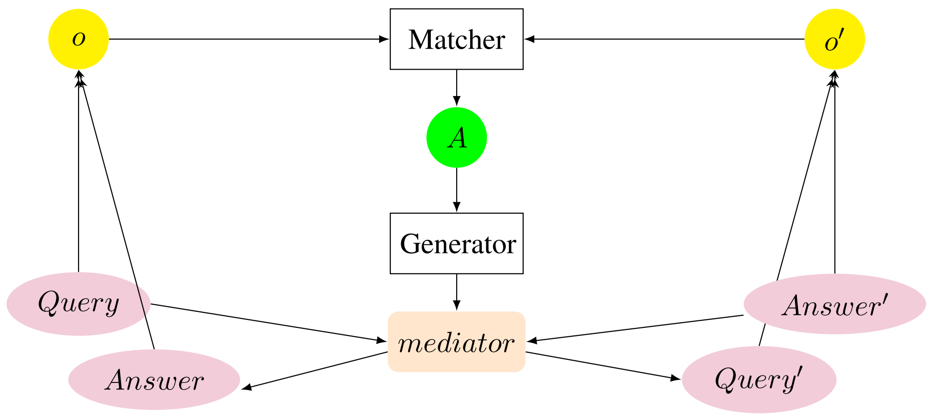


Figure 59.1: Query mediation (from [Euzenat and Shvaiko, 2007]). From two matched ontologies o and o', resulting in alignment A, a $mediator$ is generated. This allows the transformation of queries expressed with the entities of the first ontology into a query using the corresponding entities of a matched ontology and the translation back of the results from the second ontology to the first one.

Translating the answers requires the possibility of inverting the alignments (Invert operator). The generated functions should be compatible, otherwise the translated answer may not be a valid answer to the initial query. Compatibility can be expressed as follows:

$$\forall e \in o, \mathsf{TransformQuery}(\mathsf{TransformQuery}(e, A), \mathsf{Invert}(A)) \sqsubseteq e$$

Here we use a subsumption relation ($\sqsubseteq$), but it can be replaced by any suitable relation ensuring that the answer is compatible.

However, it is not always necessary to translate the answers, since they can be objects independent from the ontologies, e.g., URI, files, strings.

59.2 Systems

Query mediation is mainly used in data integration and peer-to-peer systems. When the mediator content is expressed as SQL view definitions, many database systems can process them. The Alignment API (§55.1.4) can behave as a SPARQL query mediator from simple alignments.

Some systems directly generate mediators after matching, such as Wise-Integrator [He *et al.*, 2004], DIKE [Palopoli *et al.*, 2003], Artemis [Castano *et al.*, 2000]. Clio [Haas *et al.*, 2005] can serialise alignments into different query languages, e.g., SQL, XSLT, XQuery, thus enabling query answering.

Chapter 60

Reasoning

Reasoning consists of using an alignment as axioms for reasoning with the two matched ontologies. Bridge axioms used for merging can also be viewed as such axioms.

Two kinds of reasoning can be considered. The first one reasons with the ontologies for improving them. This kind of tools provides a support for ontology matching. The second one uses the ontology and the alignments for finding their consequences, e.g., for querying in a semantic peer-to-peer system.

We present below these two types of reasoning. They are both based on the possibility, given an alignments and ontologies, to decide what are their consequences (denoted by $\models$). This assumes that the alignment is expressed in some kind of logic. For that purpose, we will consider a TransformAsRules operation which builds a set of axioms from an alignment:

$$\mathsf{TransformAsRules}(A) = o$$

Here the set of rules is represented as an ontology o which must be written in an ontology language supporting rules or the expression of bridge axioms (in OWL, C-OWL, SWRL,r F-logic, etc.).

We present below the techniques for improving alignments and reasoning with alignments.

60.1 Reasoning with alignments

Recently, several logics have been implemented providing semantics to ontology with alignments: DDL [Borgida and Serafini, 2003b], $\mathcal{E}$-connections [Grau *et al.*, 2006], equalising semantics [Zimmermann and Euzenat, 2006]. These semantics provide the consequence relations for their semantics. They are thus the necessary basis for reasoning with alignments. This consist of asking to a reasoner implementing this semantics if a particular formula is a consequence of the ontologies with alignments and can be used in various applications where an assertion has to be evaluated with regard to alignments (semantic peer-to-peer systems, agent systems).

60.2 Alignment enhancement

We distinguish here between two techniques for improving alignments: detecting inconsistencies in alignments and removing them or finding a reduced or expanded form for alignments.

60.2.1 Inconsistency recovery

Inconsistency recovery consists of detecting inconsistencies caused by alignments which are discovered by heuristic or syntax-based methods. In particular, such alignments often contain wrong and redundant correspondences. Several works [Meilicke *et al.*, 2006; Meilicke *et al.*, 2007] exploit successfully the reasoning algorithm implemented in DRAGO to improve or repair ontology alignments which are created by different matching systems without any human intervention. The proposed method can be used to check (automatically created) alignments for formal consistency and determine deduced correspondences that have not explicitly been represented. The basic assumption is that *a correspondence that correctly states the semantic relations between ontologies should not cause inconsistencies in any of the ontologies*. This means that by diagnosing inconsistencies in (matched) local ontologies, we would discover incorrect sets of correspondences to be removed.

As suggested in [Meilicke *et al.*, 2006], the improving process consists of several steps :

1. *Correspondence creation*. In order to support automatic repair of inconsistent correspondences later on (step 3), the matching algorithms chosen should ideally not only return a set of correspondences but also a level of confidence in the correctness of a correspondence.
2. *Diagnosis*. With the help of an inference engine we can identify unsatisfiable concepts, and correspondences which are responsible for any unsatisfiability. We assume that the initial ontologies do not contain unsatisfiable concepts. If we now observe an unsatisfiable concept in the target ontology this implies that is caused by some correspondences in the alignment. Considering the unsatisfiable concept as a symptom, this step then tries to identify and repair the cause of this unsatisfiability. For this purpose, an irreducible conflict set for the identified unsatisfiable concept has to be computed. An irreduccible conflict set is a set of correspondences that makes the concept unsatisfiable and removing a correspondence from this set makes the concept satisfiable again.
3. *Heuristic debugging*. From the irreducible conflict set of correspondences computed, this step will decide which correspondence to remove. For this purpose, several approaches can be used: (i) removing the correspondence whose level of confidence is the smallest (automatically), (ii) if there are several correspondences that have the same level of confidence, displaying whole conflict set and leaving the decision to the user.

60.2.2 Expanding and reducing alignments

Expanding and reducing alignments consists of computing their deductive closure or reduction defined as:

$$Cn(A) = \{\alpha; o, o', A \models \alpha\}$$

and $Red(A)$ can be defined algebraically by:

1. $Red(A) \models A$
2. $A \models Red(A)$
3. $\forall A'; A' \models A$ and $A \models A \Rightarrow A' \not\subset Red(A)$

Contrary to the deductive closure, the reduction is usually not unique. This operation may be interesting for removing redundancies from an alignment. The idea is that if a correspondence is entailed from an alignment that does not contain that correspondence, then it is logically redundant with respect to that alignment. As a consequence, such correspondences can be removed from the alignment without changing semantically the alignment. Reducing alignments by this way can be used as the last step in approaches to improving alignments. It can also be useful for presenting alignments to users in a minimal way or for more easily comparing alignments.

60.2.3 Consistent alignment merge

Merging alignments combines several available alignments between two ontologies. A consistent merge, does this so that the result is consistent by checking if they are partly or completely compatible together. If adding to an alignment A a correspondence α picked from another does not cause any inconsistency then A can be expanded by α. This task may be interesting when an alignment-based application needs a *maximal consistent alignment merge* that is built from several available alignments created by different methods. That alignment can be a candidate for a compromising solution that allows to reuse the most possible knowledge from available alignments.

Definition 126 (Expansion)**.** *Let $S = \langle o, o', \{A_1\}\rangle$ be consistent. Let A_2 be another alignment of o, o'. We say $A_1 \cup \{\alpha\}$ is an expansion of A_1 by $\alpha \in A_2$ iff $\langle o, o', \{A_1 \cup \{\alpha\}\}\rangle$ is consistent.*

Note that in Definition 126 it is not necessary that S entails α since adding α to A_1 may reduce the set of models of S. From Definition 126, we can give a formal definition for *maximal consistent alignment merge* as follows.

Definition 127 (Maximal consistent alignment merge)**.** *Let $S = \langle o, o', \mathbf{A}\rangle$ such that $\langle o, o', \{A\}\rangle$ is consistent for all $A \in \mathbf{A}$. A_M is a maximal consistent alignment merge iff*

1. *$A_M \subseteq \bigcup_{A\in\mathbf{A}} A$,*
2. *$\langle o, o', \{A_M\}\rangle$ is consistent, and*
3. *$\langle o, o', \{A_M \cup \{\alpha\}\}\rangle$ is inconsistent for all $\alpha \in \bigcup_{A\in\mathbf{A}} A \setminus A_M$.*

Conditions 1 and 2 guarantee that A_M is an admissible alignment constructed from the available alignments. Finally, the maximality of A_M is ensured by Condition 3. There may exist several maximal consistent alignment merges of $S = \langle o, o', \mathbf{A}\rangle$ since the expansion of an alignment by a correspondence as described in Definition 126 is nondeterministic.

A procedure for computing a maximal consistent alignment merge of $S = \langle o, o', \mathbf{A}\rangle$ can be directly devised from Definition 126 and 127. Such a procedure would consist of the following steps:

1. For each $A \in \mathbf{A}$,
2. For each $\alpha \in \bigcup_{A\in\mathbf{A}} A \setminus A_M$,
3. If $A_M \cup \{\alpha\}$ is consistent then $A_M := A_M \cup \{\alpha\}$. Repeat step 2.

Since alignments contain a finite number of correspondences, this procedure terminates. We can verify that A_M obtained from this procedure is a maximal consistent alignment merge according to Definition 127. However, this procedure would be particularly inneficient.

60.3 Systems

The inference engines Pellet [Sirin *et al.*, 2007] and DRAGO [Serafini and Tamilin, 2005] implement algorithms to reason on OWL ontologies with alignments. DRAGO is able to reason with the C-OWL language [Bouquet *et al.*, 2003a] while Pellet can deal with OWL "modules" based on $\mathcal{E}$-connection.

More generaly, it is possible to retain the unique domain semantics and consider using the merge of both ontologies with a representation of the alignments for reasoning with them. Any transformation of the alignments under a form suitable for reasoning, such as SWRL, OWL, or F-Logic can be used by inference engines for these languages, such as Pellet [Sirin *et al.*, 2007], FaCT++, Racer, or Flora. The Alignment API can transform simple alignments into set of such rules. In OntoMerge (mentioned in §56), once the merged ontology is constructed, inferences can be carried out either in a demand-driven (backward-chaining) or data-driven (forward chaining) way with the help of a first-order theorem prover, called *OntoEngine*.

The inconsistency recovery technique has also been implemented in the ASMOV system [Jean-Mary and Kabuka, 2007].

Chapter 61

Run time services for storing and sharing alignments

There are several reasons why applications using ontology matching could benefit from sharing ontology matching techniques and results:

– *Each application can benefit from more algorithms*: Many different applications have comparable needs. It is thus appropriate to share the solutions to these problems. This is especially true as alignments are quite difficult to provide.
– *Each algorithm can be used in more applications*: Alignments can be used for different purposes and must be expressed as such instead of as bridge axioms, mediators or translation functions.
– *Each individual alignment can be reused by different applications*: There is no magic algorithm for quickly providing a useful alignment. Once high quality alignments have been established – either automatically or manually – it is very important to be able to store, share and reuse them.

For that purpose, it is useful to provide an alignment service able to store and share existing alignments as well as to generate new alignments on-the-fly. This kind of service should be shared by the applications using ontologies on the semantic web. They should be seen as a directory or a service by web services, as an agent by agents, as a library in ambient computing applications, etc. Operations that are necessary in such a service include:

– the ability to store alignments and retrieve them, disregarding whether they are provided by automatic means or manually;
– the proper annotation of alignments in order for the clients to evaluate the opportunity to use one of them or to start from it (this starts with the information about the matching algorithms and the justifications for correspondences that can be used in agent argumentation);
– the ability to produce alignments on-the-fly through various algorithms that can be extended and tuned;
– the ability to generate knowledge processors, such as mediators, transformations, translators and rules as well as to run these processors if necessary;
– the possibility to discover similar ontologies and to interact with other such services in order to ask them for operations that the current service cannot provide by itself.

Such a service would require a standardisation support, such as the choice of an alignment format or at least of metadata format. There have been proposals for providing matching systems and alignment stores that can be considered as servers [Euzenat, 2005; Zhdanova and Shvaiko, 2006], but they need a wider availability (to agents, services, etc.) and achieving a critical mass of users to really be helpful.

Alignment support can be implemented either as a component of an ontology management tool and even being specific to each particular workstation (see Chapter 55). However, in order to optimise sharing, which is an important benefit of using alignments, it is better to store the alignments in an independent alignment server. Such a server can be either used for sharing alignments among a particular organisation or open to the semantic web at large.

61.1 Storing alignments

If alignments between widely accepted ontologies are required, they will have to be found over and over again. Hence, as mentioned in the requirements, the alignments should be stored and shared adequately. An infrastructure capable of storing the alignments and of providing them on demand to other users would be useful.

Alignment servers are independent software components which offer a library of matching methods and an alignment store that can be used by their clients. In a minimal configuration, alignment servers contribute storing and communicating alignments. Ideally, they can offers all the services identified in Chapter 54 and in particular alignment manipulation.

Alignment servers serve two purposes: for design time ontology matching, they will be components loosely coupled to the ontology management environment which may ask for alignments and for exploiting these alignments. For run time matching, the alignment servers can be invoked directly by the application. So, alignment servers will implement the services for both design time and run time matching at once.

These servers are exposed to clients, either ontology management systems or applications, through various communication channels (agent communication messages, web services) so that all clients can effectively share the infrastructure. A server may be seen as a directory or a service by web services, as an agent by agents, as a library in ambient computing applications, etc.

Alignment servers must be found on the semantic web. For that purpose they can be registered by service directories, e.g., UDDI for web services. Services or other agents should be able to subscribe some particular results of interest by these services. These directories are useful for other web services, agents, peers to find the alignment services.

In addition, servers can be grouped into an alignment infrastructure which supports them in communicating together. They can exchange the alignments they found and select them on various criteria. This may be useful for alignment servers to outsource some of their tasks. In particular, it may happen that:

- they cannot render an alignment in a particular format;
- they cannot process a particular matching method;
- they cannot access a particular ontology;
- a particular alignment is already stored by another server.

In these events, the concerned alignment server will be able to call other servers. This is especially useful when the client is not happy with the alignments provided by the current server, it is then possible to either deliver alignments provided by other servers or to redirect the client to these servers.

Moreover, this opens the door to value-added alignment services which use the results of other servers as a pre-processing for their own treatments or which aggregates the results of other servers in order to deliver a better alignment.

61.2 Sharing alignments

The main goal of storing alignments is to be able to share them among different applications. Because these applications have diverse needs and various selection criteria, it is necessary to be able to search and retrieve alignments on these criteria. Alignment metadata used for indexing alignments are thus very important. So far, alignments contain information about:

– the aligned ontologies;
– the language in which these ontology are expressed;
– the kind of alignment it is (1:1 or n:m for instance);
– the algorithm that provided it (or if it has been provided by hand);
– the confidence in each correspondence.

This information is already very precious and helps applications selecting the most appropriate alignments. It is thus necessary that ontology matchers be able to generate and alignment servers be able to store these metadata. Oyster [Palma and Haase, 2005], a peer-to-peer infrastructure for sharing metadata about ontologies that can be used in ontology management, has been extended for featuring some metadata about alignments.

However, metadata schemes are extensible and other valuable information may be added to alignment format, such as:

– the parameters passed to the generating algorithms;
– the properties satisfied by the correspondences (and their proof if necessary);
– the certificate from an issuing source;
– the limitations of the use of the alignment;
– the arguments in favor or against a correspondence [Laera *et al.*, 2007].

All such information can be useful for evaluating and selecting alignments and thus should be available from alignment servers.

61.3 The Alignment server

The Alignment server has been proposed in [Euzenat, 2005] and implemented as reported in [Euzenat *et al.*, 2007] in order to suit the purpose of storing and sharing alignments and methods for finding alignments. Such a server enables matching ontologies, storing the resulting alignment, storing manually provided alignments, extracting merger, transformer, mediators from these alignments.

The Alignment Server is built around the Alignment API developed by INRIA (see Figure 61.1). It thus provides access to all the features of this API (see Table 61.1). The server ensures the persistence of the alignments through the storage of these in a relational database.

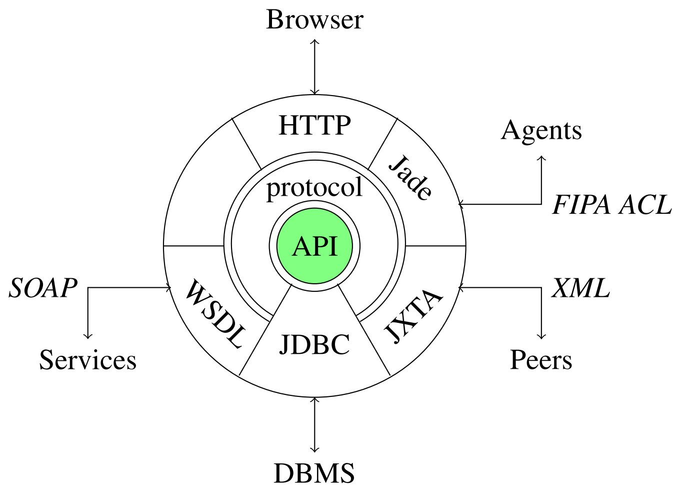


Figure 61.1: The Alignment Server is built on the Alignment API that is seated on top of a relational database repository for alignment and is wrapped around a simple protocol. Each access method is a plug-in that interacts with the server through the protocol. Currently, HTML, agent and web service plug-ins are available.

The server is available for applications at design time and/or at run time. There is no constraint that the alignments are computed on-line or off-line, i.e., they are stored in the alignment store, or that they are processed by hand or automatically.

Service	Syntax
Finding a similar ontology	$o' \Leftarrow Match(o, t)$
Align two ontologies	$A' \Leftarrow Align(o, o', A, p)$
Thresholding	$A' \Leftarrow Threshold(A, V)$
Generating code	$P \Leftarrow Render(A, language)$
Translating a message	$m' \Leftarrow Translate(m, A)$
Storing alignment	$n \Leftarrow Store(A, o, o')$
Suppressing alignment	$Delete(n)$
Finding (stored) alignments	$\{n\} \Leftarrow Find(o, o')$
Retrieving alignment	$\langle o, o', A\rangle \Leftarrow Retrieve(n)$

Table 61.1: Services provided by the alignment service and corresponding API primitives (o denotes an ontology, A an alignment, p parameters, n an index denoting an alignment, P a program realising the alignment and t and m some expressions, namely, terms to be matched and messages to be translated).

Access to the API is achieved through a protocol which extends the one designed in [Euzenat *et al.*, 2005a]. Plug-ins allow the remote invocation of the alignment server (see Figure 61.1). At the time of writting, three plug-ins are available for the server:

– HTTP/HTML plug-in for interacting through a browser;

- JADE/FIPA ACL for interacting with agents;
- HTTP/SOAP plug-in for interacting as a web service.

The components of the Alignment Server as well as the connected clients can be distributed in different machines. Several servers can share the same databases (the server works in write once mode: it never modifies an alignment but always creates new ones; not all the created alignments being stored in the database *in fine*). Applications can reach the Alignment server by any way they want, e.g., starting by using Jade and then turning to web service interface.

Alignment services must be found on the semantic web. For that purpose they can be registered by service directories, e.g., UDDI for web services. These directories are useful for other web services, agents, peers to find the alignment services. They are even more useful for alignment services to basically outsource some of their tasks. In particular, it may happen that:

- they cannot render an alignment in a particular format;
- they cannot process a particular matching method;
- they cannot access a particular ontology;
- a particular alignment is already stored by another service.

In these events, the concerned alignment service will be able to call other alignment services. This is especially useful when the client is not happy with the alignments provided by the current service, it is then possible to either deliver alignments provided by other services or to redirect the client to these services.

Like the Alignment API, the Alignment server can always be extended. In particular, it is possible to add new matching algorithms and mediator generators that will be accessible through the API. They will also be accessible through the alignment services. Services can thus be extended to new needs without breaking the infrastructure.

Chapter 62

Conclusion

Dealing with ontology heterogeneity involves finding the alignments, or sets of correspondences, existing between ontology entities and processing them for reconciling the ontologies. We have reviewed techniques for processing alignments and systems implementing these techniques. Alignments can be used in different ways (merging, transformation, translation, mediation) and there are different languages adapted to each of these ways (SWRL, OWL, C-OWL, XSLT, SQL, etc.). Beyond alignment processing, we also have identified the need for alignment management at both design time and run time.

So far there are only a few systems able to process alignments in such diverse ways. Several matching systems process directly their results in one of these operation, while others deliver alignments. Unfortunately, most often, the delivered alignments are in a format that cannot be exploited by other systems and operator generators, thus requiring additional efforts to embed them into the new environments. This is not particularly useful for alignment management.

Useful alignments are such a scarce resource that storing them in an independent format such as those seen in deliverable D2.2.6 and D2.2.10 is very important. It would allow sharing and processing them in different ways independently form the applications. This would give more freedom to application developers to choose the best suited algorithm and to process alignments adequately.

The small number of systems implementing these techniques with regard to the large number of systems for ontology matching validate the two-steps approach for dealing with heterogeneity used in Knowledge web. No system implements all the proposed features, though those implemented by Knowledge web partners, namely the Alignment API from INRIA and WSMT from the University of Innsbruck, are the closest to this, by implementing the two-steps approach.

Alignment management is not as advanced as ontology management and much remains to be developed for fully supporting and sharing alignments on a wide scale. Challenges for alignment management include adoption challenges and research problems. The important challenge is to have a natural integration of alignment management with most of the ontology engineering and ontology management systems. If alignment sharing and management is to become a reality, then there should not be one proprietary format with each tool that cannot be handled by other tools. Knowledge web has worked towards this by providing expressive formats for alignments. Another challenge is the easy retrieving of available alignments. To some extent, we pursue this effort within other projects (e.g., NeOn). For this purpose, proper alignment metadata and web-wide search support have to be set up.

There remains difficult research problems in the domain of alignment management such as:

– The identification of duplicate alignments or evolutions from a particular alignment;
– Aggregating, composing and reasoning usefully with a massive number of alignments;
– The design of ever better user interaction systems for both interacting with matching systems and editing alignments.

Acknowledgements

We appreciate technical feedback from Mark Ehrig, Stefano Zanobini and Antoine Zimmermann on the material of Chapter 23.

We thank Songmao Zhang (Chinese Academy of Sciences) and Olivier Bodenreider (U.S. National Library of Medicine) for allowing us to reuse some material of [Zhang and Bodenreider, 2007b] concerning the cross-validation of the alignments in the anatomy test case of Chapter 34.

We warmly thank each participant of the OAEI-2006 campaign. We know that they have worked hard for obtaining their results. We are grateful to the teams of the University of Washington and the University of Manchester for allowing us to use their ontologies of anatomy. Finally, we thank the other members of the Ontology Alignment Evaluation Initiative Steering committee for the support in the organization of the OAEI-2006 campaign: Benjamin Ashpole (Lockheed Martin Advanced Technology Lab.), Lewis Hart (AT&T, USA), Tadashi Hoshiai (Fujitsu, Japan), Todd Hughes (DARPA, USA), Yannis Kalfoglou (University of Southampton, UK), John Li (Teknowledge, USA), Natasha Noy (Stanford University, USA), York Sure (University of Karlsruhe, Germany) Raphal Troncy (CWI, Amsterdam, The Netherlands), and Petko Valtchev (Universit du Qubec Montral, Canada).

We also thank all the providers of test sets that have been used in the do2002a: the University of Washington, the University of Manchester Google, Yahoo and Looksmart, the Food and Agriculture Organization of the United Nations, the United States National Agricultural Library, and World Wide Jobs GmbH.

Appendix A

Overview of processing systems

Here are the systems reviewed in [Euzenat and Shvaiko, 2007] which can process alignments (operation).

Table A.1: Various ontology matching systems offering alignment processing.

System	Input	Needs	Output	Operation
TransScm [Milo and Zohar, 1998]	SGML, OO	Semi	Translator	Data translation
SKAT [Mitra *et al.*, 1999]	RDF	Semi	Bridge rules	Data translation
COMA & COMA++ [Do, 2005]	Relational schema, XML schema, OWL	User	Alignment	Data translation
ToMAS [Velegrakis *et al.*, 2003]	Relational schema, XML schema	Query, Alignment	Query, Alignment	Data translation
MapOnto [An *et al.*, 2006]	Relational schema, XML schema, OWL	Alignment	Rules	Data translation
sPLMap [Nottelmann and Straccia, 2005]	Database schema	Auto, Instances, Training	Rules	Data translation
Clio [Miller *et al.*, 2000]	Relational schema, XML schema	Semi, Instances (opt.)	Query transformation	Data translation
DIKE [Palopoli *et al.*, 2003]	ER	Semi	Merge	Query mediation
Artemis [Castano *et al.*, 2000]	Relational schema, OO, ER	Auto	Views	Query mediation
oMap [Straccia and Troncy, 2005b]	OWL	Auto, Instances (opt.), Training	Alignment	Query answering
H-Match [Castano *et al.*, 2006b]	OWL	Auto	Alignment	P2P query mediation
Tess [Lerner, 2000]	Database schema	Auto	Rules	Version matching
OntoBuilder [Modica *et al.*, 2001]	Web form, XML schema	User	Mediator	Ontology merging
Anchor-Prompt [Noy and Musen, 2001]	OWL, RDF	User	Axioms (OWL/RDF)	Ontology merging
OntoMerge	OWL	Alignment	Ontology	Ontology merging
HCONE [Kotis *et al.*, 2006]	OWL	Auto, Semi, User	Ontology	Ontology merging
FCA-merge	Ontology	User,	Ontology	Ontology merging

Table A.1: Various ontology matching systems offering alignment processing (continued).

System	Input	Needs	Output	Operation
[Stumme and Mädche, 2001]		Instances		
DCM	Web form	Auto	Alignment	Data integration
[He and Chang, 2006]				
Wang & al.	Web form	Instances	Alignment	Data integration
Wise-Integrator	Web form	Auto	Mediator	Data integration

Appendix B

Semantic chart

This appendix summarises the abstract syntax and semantics of the expressive alignment language. It is dependent upon:

– an external type system providing functions $D(\cdot)$, $L(\cdot)$ and $L2V(\cdot)$ mapping values to a set $\mathbb{D}$ (§50.3.1);
– an external set of operators providing comparators g_{cp} and transformations h_{transf} (§50.3.1);
– external ontology semantics providing $\mathcal{I}_U$ mapping URI to a set Δ and instances to a set $\mathcal{D}$ (§50.4).

In case of syntactically incorrect expressions, this semantics will provide undefined results.

Abstract syntax	Interpretation	Domain
Literals (v)	$\mathcal{I}_L$	$\mathbb{D}$
$"v"\hat{}\hat{}d$		$L2V(D(d))(v)$
$"v"$		$"v"$
URI References (u)	$\mathcal{I}_U$	$\Delta \supseteq \mathcal{D} \cup 2^{\mathcal{D}} \cup 2^{\mathcal{D}\times(\mathcal{D}\cup\mathbb{D})}$
u		$u^{\mathcal{I}_U}$
Paths (Q)	$\mathcal{I}_P$	$2^{\mathcal{D}\times(\mathcal{D}\cup\mathbb{D})}$
p		$p^{\mathcal{I}_U}$
r		$r^{\mathcal{I}_U}$
$Q.p$		$\{\langle x,y\rangle \in \mathcal{D}\times\mathbb{D}/\exists z \in \mathcal{D}/\langle x,z\rangle \in Q^{\mathcal{I}_P} \wedge \langle z,y\rangle \in p^{\mathcal{I}_P}\}$
$Q.r$		$\{\langle x,y\rangle \in \mathcal{D}\times\mathcal{D}/\exists z \in \mathcal{D}/\langle x,z\rangle \in Q^{\mathcal{I}_P} \wedge \langle z,y\rangle \in r^{\mathcal{I}_P}\}$
ϵ		$\{\langle x,x\rangle/x \in \mathcal{D}\}$
Paths or values (V)	$\mathcal{I}_V$	$2^{\mathcal{D}\times(\mathcal{D}\cup\mathbb{D})}$
v		$\mathcal{D}\times\{v^{\mathcal{I}_L}\}$
i		$\mathcal{D}\times\{i^{\mathcal{I}_U}\}$
Q		$Q^{\mathcal{I}_P}$
$\mathrm{transf}(V_1\ldots V_k)$		$\{\langle x, h_{\mathrm{transf}}(y_1,\ldots,y_n)\rangle/\langle x,y_1\rangle \in V_1^{\mathcal{I}_V} \wedge\cdots\wedge \langle x,y_n\rangle \in V_k^{\mathcal{I}_V}\}$
Restrictions (K)	$\mathcal{I}_K$	$2^{\mathcal{D}}$
$Q\ cp\ V$		$\{x \in \mathcal{D}/\exists y,y' \in \mathcal{D}\cup\mathbb{D}/\langle x,y\rangle \in Q^{\mathcal{I}_P} \wedge \langle x,y'\rangle \in V^{\mathcal{I}_V} \wedge g_{\mathrm{cp}}(y,y')\}$
$Q\ cp\ d$		$\{x \in \mathcal{D}/\forall y \in \mathcal{D}\cup\mathbb{D}, (\langle x,y\rangle \in Q^{\mathcal{I}_P}) \Rightarrow g_{\mathrm{cp}}(y, L(D(d)))\}$
$\lvert Q\rvert\ cp\ n$		$\{x \in \mathcal{D}/g_{\mathrm{cp}}(\lvert\{y \in \mathcal{D}\cup\mathbb{D}/\langle x,y\rangle \in Q^{\mathcal{I}_P}\}\rvert, n^{\mathcal{I}_L})\}$

Table B.1: Abstract syntax and semantics.

Abstract syntax	Interpretation	Domain
Classes (C)	$\mathcal{I}_C$	$2^{\mathcal{D}}$
c		$c^{\mathcal{I}_U}$
$C \sqcup C'$		$C^{\mathcal{I}_C} \cup C'^{\mathcal{I}_C}$
$C \sqcap C'$		$C^{\mathcal{I}_C} \cap C'^{\mathcal{I}_C}$
$\neg\mathcal{C}$		$\mathcal{D} \setminus C^{\mathcal{I}_C}$
$\exists K$		$K^{\mathcal{I}_K}$
Relations (R)	$\mathcal{I}_{CP}$	$2^{\mathcal{D}\times\mathcal{D}}$
r		$r^{\mathcal{I}_U}$
$R \sqcup R'$		$R^{\mathcal{I}_{CP}} \cup R'^{\mathcal{I}_{CP}}$
$R \sqcap R'$		$R^{\mathcal{I}_{CP}} \cap R'^{\mathcal{I}_{CP}}$
$\neg R$		$\mathcal{D} \times \mathcal{D} \setminus R^{\mathcal{I}_{CP}}$
$\text{dom}(C)$		$\{\langle x, y\rangle \in \mathcal{D} \times \mathcal{D} / x \in C^{\mathcal{I}_C}\}$
$\text{range}(C)$		$\{\langle x, y\rangle \in \mathcal{D} \times \mathcal{D} / y \in C^{\mathcal{I}_C}\}$
$\text{inv}(R)$		$\{\langle x, y\rangle / \langle y, x\rangle \in R^{\mathcal{I}_{CP}}\}$
$\text{sym}(R)$		$R^{\mathcal{I}_{CP}} \cup \text{inv}(R)^{\mathcal{I}_{CP}}$
$\text{trans}(R)$		$R^{\mathcal{I}_{CP}} \cup \{\langle x, z\rangle / \exists y \in \mathcal{D}, \langle x, y\rangle \in R^{\mathcal{I}_{CP}} \wedge \langle y, z\rangle \in trans(R)^{\mathcal{I}_{CP}}\}$
$\text{refl}(R)$		$R^{\mathcal{I}_{CP}} \cup \{\langle x, x\rangle / x \in \mathcal{D}\}$
Properties (P)	$\mathcal{I}_{DP}$	$2^{\mathcal{D}\times\mathbb{D}}$
p		$p^{\mathcal{I}_U}$
$P \sqcup P'$		$P^{\mathcal{I}_{DP}} \cup P'^{\mathcal{I}_{DP}}$
$P \sqcap P'$		$P^{\mathcal{I}_{DP}} \cap P'^{\mathcal{I}_{DP}}$
$\neg P$		$\mathcal{D} \times \mathbb{D} \setminus P^{\mathcal{I}_{DP}}$
$\text{dom}(C)$		$\{\langle x, y\rangle \in \mathcal{D} \times \mathbb{D} / x \in C^{\mathcal{I}_C}\}$
$\text{range}(d)$		$\{\langle x, y\rangle \in \mathcal{D} \times \mathbb{D} / y \in L(D(d))\}$
Instances (i)	$\mathcal{I}_U$	$\mathcal{D}$
i		$i^{\mathcal{I}_U}$
Expressions (E)	$\mathcal{I}$	$\Delta \supseteq \mathcal{D} \cup 2^{\mathcal{D}} \cup 2^{\mathcal{D}\times(\mathcal{D}\cup\mathbb{D})}$
C		$C^{\mathcal{I}_C}$
R		$R^{\mathcal{I}_{CP}}$
P		$P^{\mathcal{I}_{DP}}$
i		$i^{\mathcal{I}_U}$

Table B.1: Abstract syntax and semantics.

Appendix C

Data manipulation

C.1 Data operator table

Type	Id	Origin	explanation
numeric	add	XQuery	Returns the arithmetic sum of the first argument through the second argument.
numeric	subtract	XQuery	Returns the arithmetic difference of the first argument minus the second argument.
numeric	multiply	XQuery	Returns the arithmetic product of the first argument by the second argument.
numeric	divide	XQuery	Returns the arithmetic quotient of the first argument over the second argument.
numeric	integer-divide	XQuery	Returns the integer part of the arithmetic quotient of the first argument over the second argument.
numeric	mod	XQuery	Returns the modulo of the arithmetic quotient of the first argument over the second argument.
numeric	pow		Returns the first argument raised to the second argument power.
numeric	unary-minus	XQuery	Returns its argument with the sign changed.
string	concat	XQuery	Concatenates two strings.
string	substring	XQuery	Returns the substring of its first argument starting at the position denoted by its second argument and ending at the one denoted by the third argument.
string	length	XQuery	Returns the integer corresponding to the number of characters of the string in argument.
string	normalize-space	XQuery	Returns the whitespace-normalised value of the string in argument.
string	upper-case	XQuery	Returns the upper-cased value of the string in argument.
string	lower-case	XQuery	Returns the lower-cased value of the string in argument.
string	translate	XPath/XQuery	Returns its first string argument with occurrences of characters contained in the second argument replaced by the character at the corresponding position in the string of the third argument.

Table C.1: Operators.

Type	Id	Origin	explanation
string	replace	XQuery	Returns its first string argument with every substring matched by the regular expression in the second argument replaced by the replacement string of the third argument.
string	tokenize	XQuery	Returns a sequence of strings whose values are ordered substrings of the first argument separated by substrings that match the regular expression the second argument.
uri	resolveURI	XQuery	Returns the URI reference value of its argument resolved.
collection	concatenate	XQuery	Returns the concatenation of its list arguments.
collection	intersection		Returns a list containing elements found in both the first list argument and the second list argument.
collection	union		Returns a list containing the elements found in any of its list arguments.
collection	difference		Returns a list containing the elements of the first list argument that are not members of the second list argument.
integer	length	Lisp	Returns the number of elements in its list argument.

Table C.1: Operators.

C.2 Comparator table

Type	Id	Origin	explanation
all	equal	XQuery	Satisfied iff the first argument and the second argument are the same.
all	not-equal	SWRL	The negation of equal.
ordered	less-than	XQuery	Satisfied iff the first argument and the second argument are both in some implemented type and the first argument is less than the second argument according to a type-specific ordering (partial or total), if there is one defined for the type. The ordering function for the type of untyped literals is the partial order defined as string ordering when the language tags are the same (or both missing) and incomparable otherwise.
ordered	less-than-or-equal	SWRL	Either less than, as above, or equal, as above.
ordered	greater-than	XQuery	Similarto less-than.
ordered	greater-than-or-equal	SWRL	Similar to less-than-or-equal.
string	contains	XQuery	Satisfied iff the first argument contains the second argument (case sensitive)
string	starts-with	XQuery	Satisfied iff the first argument starts with the second argument.
string	ends-with	XQuery	Satisfied iff the first argument ends with the second argument.
string	matches	XQuery	Satisfied iff the first argument matches the regular expression in the second argument.
collection	contains	XQuery	Satisfied iff the first argument contains the second argument
collection	includes	XQuery	Satisfied iff the first argument contains all the elements the second argument.
collection	includes-strictly	XQuery	Satisfied iff the first argument contains more elements than the the second argument.
collection	empty		Satisfied iff its first list argument is an empty list.

Table C.2: Comparators.

Appendix D

OWL Ontology

This appendix exhibits the alignment exchange language as a OWL ontology (version of 17/03/2007) that can be found at `http://www.omwg.org/TR/d7/ontology/alignment/`.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:align="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns="http://www.omwg.org/TR/d7/d7.2/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://www.omwg.org/TR/d7/d7.2/">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/PoV">
    <owl:equivalentClass>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Path"/>
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Value"/>
        </owl:unionOf>

      </owl:Class>
    </owl:equivalentClass>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A path or a value</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Cell">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>

          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/entity2"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>

        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/entity1"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>

        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/relation"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>

          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/measure"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Cells are the objects representing mapping rules</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/AttributeCondition">

    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Condition"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Set of conditions that can be applied on Attribute entities</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Instance">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Instance entities represent instance data of the ontology.</rdfs:comment>
    <rdfs:subClassOf>

      <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Entity"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/AttributeOccurenceCondition">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/ClassCondition"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This set a condition on the existence (in term of instanciation) of one of the class attri
```

```
</owl:Class>
<owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Class">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/condition"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>

    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Entity"/>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Class entities represent the objects of the ontology.</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Formalism">

  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/name"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>

  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/uri"/>
      </owl:onProperty>
    </owl:Restriction>

  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Entity">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>

        <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/operator"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >Entities are ontological entities part of the mapping rules (cells)</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Ontology">
    <rdfs:subClassOf>

      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/formalism"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>

    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Ontologies are of this type</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Condition">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The class Condition groups the various conditions that can be applied on entities. It is p
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Value">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>

      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >1</owl:cardinality>
            <owl:onProperty>
              <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/stringValue"/>
            </owl:onProperty>
          </owl:Restriction>

          <owl:Restriction>
            <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >1</owl:cardinality>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/instanceValue"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:unionOf>
      </owl:Class>

    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Value object can be instanciated either with an instanceValue or a stringValue</rdfs:com
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/ClassCondition">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/restriction"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/Condition"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Set of conditions that can be applied on Class entities</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Restriction">

    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/onProperty"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>

    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/value"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>

    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/comparator"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>

    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Restrictions objects are of this type</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Alignment">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/type"/>

        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
```

```
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:maxCardinality>

    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/xml"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:maxCardinality>

    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/method"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:minCardinality>

    <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/map"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Alignment objects represent an alignment between two ontologies. Each alignment is alos co
<rdfs:subClassOf>
  <owl:Restriction>

    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:maxCardinality>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/level"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>

    <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/onto1"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
```

```
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/onto2"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>

        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/purpose"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Attribute">

    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/Entity"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/condition"/>
        </owl:onProperty>
      </owl:Restriction>

    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Attribute enties represent properties of the ontology which range is a datatype.</rdfs:com
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/TypeCondition">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This set a condition on the atribute type.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/AttributeCondition"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/AttributeValueCondition">

    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/ClassCondition"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This set a condition on the value of one of the class attributes</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/ValueCondition">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>

          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/comparator"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This set a condition on the attribute value.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/AttributeCondition"/>
    <rdfs:subClassOf>
      <owl:Restriction>

        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/value"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/RelationCondition">

    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Set of conditions that can be applied on Relation entities</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/Condition"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Path">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Pathes objects represent pathes through the RDF graph in order to reach a particular set o
Example: The set of persons who have a female sibbling having exactly two children.</rdfs:comme
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>

        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/first"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>

        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/next"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Collection">

    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >2</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/item"/>
        </owl:onProperty>
      </owl:Restriction>

    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/AttributeTypeCondition">
    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/ClassCondition"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This set a condition on the type of one of the class attributes.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Relation">
    <rdfs:subClassOf>

      <owl:Restriction>
        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/condition"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Relation entities represent properties of the ontology which range is a Class entity. Only

    <rdfs:subClassOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/Entity"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/operator">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Entity"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >operators to combine entities</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/onto1">
    <rdfs:subPropertyOf>

      <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/ontology"/>
    </rdfs:subPropertyOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >First ontology of the alignment</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/instanceValue">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Value"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A value given as an instance entity</rdfs:comment>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Instance"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/entity">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Cell"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Entities of a Cell</rdfs:comment>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Entity"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/next">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Path"/>

    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Path"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicate the following path of a path, or nil to terminate a path</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/transitive">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >transitive closure of a relation</rdfs:comment>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/onProperty">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Restriction"/>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Path"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >properties applied to pathes</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/formalism">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicate the formalism (ontology language) of the aligned ontologies</rdfs:comment>

    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Formalism"/>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Ontology"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/inverse">
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Inverse of a relation</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/onto2">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/ontology"/>
    </rdfs:subPropertyOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Second ontology of the alignment</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/rangeRestriction">

    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Class"/>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Restrict the range of the Relation to a given Class expression</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/entity1">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >First entity of a cell</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/entity"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/item">
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Entity"/>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Collection"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/condition">
```

```
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Entity"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Define a condition on an expression in order to restrict its scope.</rdfs:comment>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Condition"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/restriction">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Condition"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A restriction specifies a the set of entities that will fulfill the related condition.</rd
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Restriction"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/or">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >cunjunction of the entities</rdfs:comment>

    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Collection"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/reflexive">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Reflexive of a relation</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/map">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Relate an alignment to each of its cells</rdfs:comment>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Cell"/>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Alignment"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/domainRestriction">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Restrict the domain of an Attribute or Relation to the given Class expression</rdfs:commen

    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/RelationCondition"/>
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/AttributeCondition"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Class"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/and">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >intersection of the entities</rdfs:comment>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Collection"/>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/symmetric">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Relation"/>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Symmetric of a relation</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/ontology">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Alignment"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Represent the source and target ontologies of an alignment</rdfs:comment>
    <rdfs:range rdf:resource="http://www.omwg.org/TR/d7/d7.2/Ontology"/>

  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/not">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Negation of an entity</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/operator"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/first">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Path"/>
    <rdfs:range>

      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Attribute"/>
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Relation"/>
          <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
              <Entity rdf:about="http://www.omwg.org/TR/d7/d7.2/null-entity">
                <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >This entity is used to terminate a path.</rdfs:comment>

              </Entity>
            </owl:oneOf>
          </owl:Class>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicates the first element in a path</rdfs:comment>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/entity2">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Second entity of a cell</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://www.omwg.org/TR/d7/d7.2/entity"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/xml">
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">

          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >no</rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >yes</rdf:first>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>

    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicate if the alignment is expressed in XML format. (deprecated?)</rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/relation">
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Cell"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >relation between the aligned entities in a mapping rule.</rdfs:comment>
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">

          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >equivalence</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >subsumption</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:parseType="Resource">
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >disjoint</rdf:first>

              </rdf:rest>
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >instanceOf</rdf:first>
            </rdf:rest>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/uri">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >uri of a formalism (ontology language)</rdfs:comment>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Formalism"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/value">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>

      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/ValueCondition"/>
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Restriction"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
```

```
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A value indicate the value compared in a condition. It can take either the value of an Att

  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/measure">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Measure of confidence in a mapping rule.</rdfs:comment>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Cell"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/purpose">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Describe the purpose of the alignment</rdfs:comment>

  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/method">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicates the method, the name of the tool or algoriithm that outputted the alignment</rdf
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/stringValue">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A value given as a string</rdfs:comment>

  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/comparator">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Comparators are used in conditions in order to make comparisons of attribute values. The v
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/Restriction"/>
          <owl:Class rdf:about="http://www.omwg.org/TR/d7/d7.2/ValueCondition"/>

        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/level">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicates the level of the alignment (deprecated?, alway 2)</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>

  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/type">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Indicate the arity of the alignment (Deprecated, always **)</rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://www.omwg.org/TR/d7/d7.2/name">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Name of a formalism or ontology langage</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="http://www.omwg.org/TR/d7/d7.2/Formalism"/>
  </owl:DatatypeProperty>
  <rdf:Description rdf:about="http://knowledgeweb.semanticweb.org/heterogeneity/alignment">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >RDF vocabulary for the alignment format</rdfs:comment>
  </rdf:Description>
  <Path rdf:about="http://www.omwg.org/TR/d7/d7.2/nil">
    <first rdf:resource="http://www.omwg.org/TR/d7/d7.2/null-entity"/>
    <next rdf:resource="http://www.omwg.org/TR/d7/d7.2/nil"/>

    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Represent a null path. Used to end a path</rdfs:comment>
  </Path>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.2, Build 311)  http://protege.stanford.edu -->
```

Bibliography

[Aberer *et al.*, 2003a] Karl Aberer, Philippe Cudré-Mauroux, A. Datta, Z. Despotovic, Manfred Hauswirth, M. Punceva, and R. Schmidt. P-Grid: A self-organizing structured p2p system. *ACM SIGMOD Record*, 32(3), 2003.

[Aberer *et al.*, 2003b] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. Start making sense: The chatty web approach for global semantic agreements. *Journal of Web Semantics*, 1(1), December 2003.

[Aberer *et al.*, 2004a] Karl Aberer, Tiziana Catarci, Philippe Cudré-Mauroux, T. Dillon, S. Grimm, Mohand-Saïd Hacid, A. Illarramendi, Mustafa Jarrar, V. Kashyap, M. Mecella, E. Mena, E. Neuhold, A. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. De Santis, Stefano Spaccapietra, Steffen Staab, Rudi Studer, and O. De Troyer. Emergent semantics systems. In Mokrane Bouzeghoub, Carole Goble, V. Kashyap, and Stefano Spaccapietra, editors, *proceeding of International Conference on Semantics of a Networked World*, LNCS, pages 14 – 44. Springer Verlag, 2004.

[Aberer *et al.*, 2004b] Karl Aberer, Philippe Cudré-Mauroux, M. Ouksel, Tiziana Catarci, Mohand-Saïd Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. Neuhold, O. De Troyer, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, Stefano Spaccapietra, Steffen Staab, and Rudi Studer. Emergent semantics principles and issues. In *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, Jeju Island, Korea, 2004.

[Abiteboul and Duschka, 1998] Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialised views. In *Proc. of the 17th ACM Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[Adámek *et al.*, 2004] Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categorie (The Joy of Cats)*. John Wiley and Sons, New-York (NY US), 2004. Corrected version of the 1990 book of the same name, available online at `http://katmat.math.uni-bremen.de/acc/`.

[An *et al.*, 2005a] Yuan An, Alexander Borgida, and John Mylopoulos. Constructing complex semantic mappings between XML data and ontologies. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 6–20, Galway (IE), 2005.

[An *et al.*, 2005b] Yuan An, Alexander Borgida, and John Mylopoulos. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In

Proc. 4th International Conference on Ontologies, Databases and Applications of Semantics (ODBASE), volume 3761 of *Lecture notes in computer science*, pages 1152–1169, Agia Napa (CY), 2005.

[An *et al.*, 2006] Yuan An, Alexander Borgida, and John Mylopoulos. Discovering the semantics of relational tables through mappings. *Journal on Data Semantics*, VII:1–32, 2006.

[Angele and Lausen, 2004] J. Angele and G. Lausen. Ontologies in F-logic. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.

[Antoniou and van Harmelen, 2004] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 67–92. Springer, 2004.

[Atzeni *et al.*, 2005] Paolo Atzeni, Paolo Cappellari, and Philip Bernstein. Modelgen: Model independent schema translation. In *Proc. 21st International Conference on Data Engineering (ICDE)*, pages 1111–1112, Tokyo (JP), 2005.

[Atzeni *et al.*, 2006] Paolo Atzeni, Paolo Cappellari, and Philip Bernstein. Model-independent schema and data translation. In *Proc. 10th Conference on Extending Database Technology (EDBT)*, volume 3896 of *Lecture notes in computer science*, pages 368–385, München (DE), 2006.

[Avesani *et al.*, 2005] Paolo Avesani, Fausto Giunchiglia, and Mikalai Yatskevich. A large scale taxonomy mapping evaluation. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 67–81, Galway (IE), 2005.

[Avesani, 2002] Paolo Avesani. Evaluation framework for local ontologies interoperability. In *AAAI Workshop on Meaning Negotiation*, 2002.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The description logic handbook: theory, implementations and applications*. Cambridge University Press, 2003.

[Bach *et al.*, 2004] Than-Le Bach, Rose Dieng-Kuntz, and Fabien Gandon. On ontology matching problems (for building a corporate semantic web in a multi-communities organization). In *Proc. 6th International Conference on Enterprise Information Systems (ICEIS)*, pages 236–243, Porto (PT), 2004.

[Baeza-Yates and Ribeiro-Neto, 1999] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[Barrasa *et al.*, 2003] Jesus Barrasa, Oscar Corcho, and Asunsión Gómez-Pérez. Fundfinder – a case study of database-to-ontology mapping. In *Proc. ISWC Semantic integration workshop, Sanibel Island (FL US)*, 2003.

[Barthélemy and Guénoche, 1992] Jean-Pierre Barthélemy and Alain Guénoche. *Trees and proximity representations*. John Wiley & Sons, Chichester (UK), 1992.

[Barwise and Seligman, 1997] Jon Barwise and Jerry Seligman. *Information flow: the logic of distributed systems*, volume 44 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge (UK), 1997.

[Batini *et al.*, 1986] Carlo Batini, Maurizio Lenzerini, and Shamkant Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[Bechhofer *et al.*, 2001] Sean Bechhofer, Carole Goble, and Ian Horrocks. DAML+OIL is not enough. In Isabel Cruz, Stefan Decker, Jérôme Euzenat, and Deborah McGuinness, editors, *Proc. Semantic web working symposium, Stanford (CA US)*, pages 151–159, 2001.

[Bechhofer *et al.*, 2003] Sean Bechhofer, Raphael Volz, and Phillip Lord. Cooking the semantic web with the OWL API. In *Proc. 2nd International Semantic Web Conference (ISWC)*, volume 2870 of *Lecture notes in computer science*, pages 659–675, Sanibel Island (FL US), 2003.

[Beeri *et al.*, 1997] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proc. of the 16th ACM Symp. on Principles of Database Systems (PODS'97)*, pages 99–108, 1997.

[Bench-Capon and Malcolm, 1999] Trevor Bench-Capon and Grant Malcolm. Formalising ontologies and their relations. In *Database and Expert Systems Applications*, pages 250–259, 1999.

[Benerecetti *et al.*, 2000] M. Benerecetti, Paulo Bouquet, and Chiara Ghidini. Contextual reasoning distilled. *Journal of Theoretical and Experimental Artificial Intelligence*, 12(3):279–305, July 2000.

[Beneventano *et al.*, 1998] Domenico Beneventano, Sonia Bergamaschi, Stefano Lodi, and Claudio Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10(4):576–598, 1998.

[Berenji and Khedkar, 1992] H. R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. on Neural Networks*, 3(5):724–740, 1992.

[Bergamaschi *et al.*, 1998] Sonia Bergamaschi, Domenico Beneventano, Silvana Castano, and Maurizio Vincini. MOMIS: An intelligent system for the integration of semistructured and structured data. Technical Report T3-R07, Università di Modena e Reggio Emilia, Modena (IT), 1998.

[Bergamaschi *et al.*, 1999] Sonia Bergamaschi, Silvana Castano, and Maurizio Vincini. Semantic integration of semistructured and structured data sources. *ACM SIGMOD Record*, 28(1):54–59, 1999.

[Berlin and Motro, 2001] J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proceeding of CoopIS*, pages 108–122, 2001.

[Berlin and Motro, 2002] Jacob Berlin and Amihai Motro. Database schema matching using machine learning with feature selection. In *Proc. 14th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 2348 of *Lecture notes in computer science*, pages 452–466, Toronto (CA), 2002.

[Bernstein *et al.*, 2000] Philip Bernstein, Alon Halevy, and Rachel Pottinger. A vision of management of complex models. *ACM SIGMOD Record*, 29(4):55–63, 2000.

[Bhushan and Rai, 2004] N. Bhushan and K. Rai, editors. *Strategic Decision Making: Applying the Analytic Hierarchy Process*. Springer, 2004.

[Bilke and Naumann, 2005] Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *Proc. 21st International Conference on Data Engineering (ICDE)*, pages 69–80, Tokyo (JP), 2005.

[Bisson, 1992] Gilles Bisson. Learning in FOL with similarity measure. In *Proc. 10th National Conference on Artificial Intelligence (AAAI)*, pages 82–87, San-Jose (CA US), 1992.

[Bizer *et al.*, 2005] C. Bizer, R. Heese, M. Mochol, R. Oldakowski, R. Tolksdorf, and R. Eckstein. The Impact of Semantic Web Technologies on Job Recruitment Processes. In *Proc. of the 7th Internationale Tagung Wirtschaftsinformatik 2005*, pages 1367–1383, 2005.

[Borgida and Serafini, 2003a] Alex Borgida and Luciano Serafini. Distributed Description Logics: Assimilating information from peer sources. *J. Data Semantics*, pages 153–184, 2003.

[Borgida and Serafini, 2003b] Alexander Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *Journal on Data Semantics*, I:153–184, 2003.

[Borgida, 1996] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82:353–367, 1996. Research Note.

[Bouquet and Serafini, 2003] Paolo Bouquet and Luciano Serafini. On the difference between bridge rules and lifting axioms. In *Proc. 4th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT)*, volume 2680 of *Lecture notes in computer science*, pages 80–93, Stanford (CA US), 2003.

[Bouquet *et al.*, 2003a] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-OWL – contextualizing ontologies. In *Proc. 2nd International Semantic Web Conference (ISWC)*, volume 2870 of *Lecture notes in computer science*, pages 164–179, Sanibel Island (FL US), 2003.

[Bouquet *et al.*, 2003b] Paolo Bouquet, Bernardo Magnini, Luciano Serafini, and Stefano Zanobini. A SAT-based algorithm for context matching. In *Proc. 4th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT)*, volume 2680 of *Lecture notes in computer science*, pages 66–79, Stanford (CA US), 2003.

[Bouquet *et al.*, 2003c] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In *Proc. 2nd International Semantic Web Conference (ISWC)*, volume 2870 of *Lecture notes in computer science*, pages 130–145, Sanibel Island (FL US), 2003.

[Bouquet *et al.*, 2004a] Paolo Bouquet, Marc Ehrig, Jérôme Euzenat, Enrico Franconi, Pascal Hitzler, Markus Krötzsch, Luciano Serafini, Giorgos Stamou, York Sure, and Sergio Tessaris. Specification of a common framework for characterizing alignment. Deliverable D2.2.1, Knowledge web NoE, 2004.

[Bouquet *et al.*, 2004b] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, 1(1):325–343, 2004.

[Bouquet *et al.*, 2006] Paolo Bouquet, Luciano Serafini, Stefano Zanobini, and Simone Sceffer. Bootstrapping semantics on the web: meaning elicitation from schemas. In *Proc. 15th International World Wide Web Conference (WWW)*, pages 505–512, Edinburgh (UK), 2006.

[Broekstra *et al.*, 2004] Jeen Broekstra, Marc Ehrig, Peter Haase, Frank van Harmelen, Maarten Menken, Peter Mika, Bjoern Schnizler, and Ronny Siebes. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the SemPGrid 04 Workshop*, New York, May 2004.

[Calì *et al.*, 2004] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 29(2):147–163, 2004.

[Calvanese *et al.*, 1998a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[Calvanese *et al.*, 1998b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[Calvanese *et al.*, 2000a] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, 2000.

[Calvanese *et al.*, 2000b] D. Calvanese, G. De Giacomo, M. Lenzerini, and Moshe Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Sym. on Logic in Computer Science (LICS 2000)*, 2000.

[Calvanese *et al.*, 2002a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Description logics for information integration. In Anthonis Kakas and Fariba Sadri, editors, *Computational logic: logic programming and beyond, essays in honour of Robert A. Kowalski*, volume 2408 of *Lecture notes in computer science*, pages 41–60. Springer, Heidelberg (DE), 2002.

[Calvanese *et al.*, 2002b] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A framework for ontology integration. In Isabel Cruz, Stefan Decker, Jérôme Euzenat, and Deborah McGuinness, editors, *The emerging semantic web*, pages 201–214. IOS Press, Amsterdam (NL), 2002.

[Calvanese *et al.*, 2003] Diego Calvanese, Elio Damaggio, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Semantic data integration in p2p systems. In *Proc. of the VLDB International Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P-2003)*, 2003.

[Castano *et al.*, 2000] Silvana Castano, Valeria De Antonellis, and Sabrina De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, 2000.

[Castano *et al.*, 2004] S. Castano, A. Ferrara, and S. Montanelli. Methods and Techniques for Ontology-based Semantic Interoperability in Networked Enterprise Contexts. In *Proc. of the 1st CAiSE INTEROP Workshop On Enterprise Modelling and Ontologies for Interoperability (EMOI - INTEROP 2004)*, pages 261–264, June 2004.

[Castano *et al.*, 2005] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Dynamic knowledge discovery in open, distributed and multi-ontology systems: Techniques and applications. In David Taniar and Johanna Rahayu, editors, *Web semantics and ontology*, chapter 8, pages 226–258. Idea Group Publishing, Hershey (PA US), 2005.

[Castano *et al.*, 2006a] Silvana Castano, Alfio Ferrara, and Gianpaolo Messa. Results of the HMatch ontology matchmaker in OAEI 2006. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, pages 134–143, Athens (GA US), 2006.

[Castano *et al.*, 2006b] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Matching ontologies in open networked systems: Techniques and applications. *Journal on Data Semantics*, V:25–63, 2006.

[Catarci and Lenzerini, 1993] T. Catarci and M. Lenzerini. Representing and Using Interschema Knowledge in Cooperative Information Systems. *Journal of Intelligent and Cooperative Systems*, 2(4):375–398, 1993.

[Chang *et al.*, 2005] Kevin Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *Proc. 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 44–55, Asilomar (CA US), 2005.

[Clements *et al.*, 2002] P. Clements, F. Bachman, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, editors. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2002.

[Clifton *et al.*, 1997] Chris Clifton, Ed Hausman, and Arnon Rosenthal. Experience with a combined approach to attribute matching across heterogeneous databases. In *Proc. 7th IFIP Conference on Database Semantics*, pages 428–453, Leysin (CH), 1997.

[Cohen *et al.*, 2003] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Proc. KDD Workshop on Data Cleaning and Object Consolidation*, pages 73–78, Washington (DC US), 2003.

[Corby and Faron, 2002] Olivier Corby and Catherine Faron. Corese: A corporate semantic web engine. In *Proc. WWW International Workshop on Real World RDF and Semantic Web Applications*, Hawai (HA US), May 2002.

[Corby *et al.*, 2000] Olivier Corby, Rose Dieng, and Cédric Hébert. A conceptual graph model for w3c resource description framework. In *Proc. 8th International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues*, Darmstadt (DE), August 2000. Springer-Verlag.

[Corby *et al.*, 2004] Olivier Corby, Rose Dieng-Kuntz, and Catherine Faron-Zucker. Querying the semantic web with the corese search engine. In *Proc. 15th ECAI/PAIS*, Valencia (ES), August 2004. IOS Press.

[Corcho, 2004] Oscar Corcho. *A declarative approach to ontology translation with knowledge preservation*. PhD thesis, Universidad Politécnica de Madrid, Madrid (ES), 2004.

[da Silva, 2004] Nuno Alexandre Pinto da Silva. *Multi-dimensional service-oriented ontology mapping*. PhD thesis, Universidade de Trás-os-Montes e Alto Douro, Villa Real (PT), 2004.

[d'Aquin *et al.*, 2003] Mathieu d'Aquin, C. Bouthier, S. Brachais, Jean Lieber, and Amedeo Napoli. Knowledge editing and maintenance tools for a semantic portal in oncology. Rapport de recherche A03-R-162, LORIA, 2003.

[De Bo *et al.*, 2004a] J. De Bo, P. Spyns, and R. Meersman. Assisting ontology integration with existing thesauri. In *Submitted to ODBASE04*, 2004.

[De Bo *et al.*, 2004b] J. De Bo, P. Spyns, and R. Meersman. Towards a methodology for semi-automatic ontology aligning and merging. Technical Report 02, Vrije Universiteit Brussel, STARLab, 2004.

[de Bruijn and Feier., 2005] J. de Bruijn and C. Feier. Report on ontology mediation for case studies. deliverable D4.6.1, SEKT, June 2005.

[de Bruijn *et al.*, 2004] Jos de Bruijn, Douglas Foxvog, and Kerstin Zimmerman. Ontology mediation patterns library. Deliverable D4.3.1, SEKT, 2004.

[de Bruijn, 2007] Jos de Bruijn. The web service modeling language WSML. Technical Report 16.1, WSMO, 2007.

[De Leenheer and Mens, 2008] Pieter De Leenheer and Tom Mens. Ontology evolution; state of the art and future directons. In Martin Hepp, Pieter De Leenheer, Aldo De Moor, and York Sure, editors, *Ontology management: semantic web, semantic web services, and business applications*, chapter 5, pages 131–176. Springer, New-York (NY US), 2008.

[Dean and Schreiber (eds.), 2004] Mike Dean and Guus Schreiber (eds.). OWL web ontology language: reference. Recommendation, W3C, 2004. http://www.w3.org/TR/owl-ref/.

[Dhamankar *et al.*, 2004] Robin Dhamankar, Yoonkyong Lee, An-Hai Doan, Alon Halevy, and Pedro Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. 23rd International Conference on Management of Data (SIGMOD)*, pages 383–394, Paris (FR), 2004.

[Didion, 2004] John Didion. The java wordnet library, 2004. http://jwordnet.sourceforge.net/.

[Dieng and Hug, 1998a] Rose Dieng and Stefan Hug. Comparison of "personal ontologies" represented through conceptual graphs. In *Proc. 13th European Conference on Artificial Intelligence (ECAI)*, pages 341–345, Brighton (UK), 1998.

[Dieng and Hug, 1998b] Rose Dieng and Stefan Hug. Multikat, a tool for comparing knowledge from multiple experts. In *Conceptual Structures: Theory, Tools and Applications, Proc. of the 6th Int. Conference on Conceptual Structures (ICCS'98)*, Montpellier (FR), August 10-12 1998. Springer-Verlag, LNAI 1453.

[Do and Rahm, 2002] Hong-Hai Do and Erhard Rahm. COMA – a system for flexible combination of schema matching approaches. In *Proc. 28th International Conference on Very Large Data Bases (VLDB)*, pages 610–621, Hong Kong (CN), 2002.

[Do *et al.*, 2002] Hong-Hai Do, Sergei Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In *Proc. Workshop on Web, Web-Services, and Database Systems*, volume 2593 of *Lecture notes in computer science*, pages 221–237, Erfurt (DE), 2002.

[Do, 2005] Hong-Hai Do. *Schema matching and mapping-based data integration*. PhD thesis, University of Leipzig, Leipzig (DE), 2005.

[Doan *et al.*, 2001] An-Hai Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. 20th International Conference on Management of Data (SIGMOD)*, pages 509–520, Santa Barbara (CA US), 2001.

[Doan *et al.*, 2003a] An-Hai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3):279–301, 2003.

[Doan *et al.*, 2003b] An-Hai Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. *VLDB journal*, 2003.

[Doan *et al.*, 2004] An-Hai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontollogy matching: a machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook on ontologies*, chapter 18, pages 385–404. Springer Verlag, Berlin (DE), 2004.

[Doan, 2002] An-Hai Doan. *Learning to map between structured representations of data*. PhD thesis, University of Washington, Seattle (WA US), 2002. http://www.kdnuggets.com/news/2004/n01/23i.html.

[Dogac *et al.*, 2002] A. Dogac, G. Laleci, Y. Kabak, and I. Cingil. Exploitingweb service semantics: Taxonomies vs. ontologies. *IEEE DATA ENGINEERING BULLETIN*, 4, 2002.

[Donini *et al.*, 1998] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274, April 1998.

[Dou *et al.*, 2003] D. Dou, D. McDermott, and P. Qi. Ontology Translation on the Semantic Web. In *Proc. of the AInt'l Conf. on Ontologies, Databases and Applications of Semantics (ODBASE2003)*, pages 952–969, 2003.

[Dou *et al.*, 2005] Dejing Dou, Drew McDermott, and Peishen Qi. Ontology translation on the semantic web. *Journal on Data Semantics*, II:35–57, 2005.

[Dragut and Lawrence, 2004] Eduard Dragut and Ramon Lawrence. Composing mappings between schemas using a reference ontology. In *Proc. 3rd International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 3290 of *Lecture notes in computer science*, pages 783–800, Larnaca (CY), 2004.

[Duschka and Genesereth, 1997] Oliver Duschka and Michael Genesereth. Infomaster – an information integration tool. In *Proc. KI Workshop on Intelligent Information Integration*, Freiburg (DE), 1997.

[Ehrig and Euzenat, 2005] Marc Ehrig and Jérôme Euzenat. Relaxed precision and recall for ontology matching. In *Proc. K-CAP Workshop on Integrating Ontologies*, pages 25–32, Banff (CA), 2005.

[Ehrig and Staab, 2004] Marc Ehrig and Steffen Staab. QOM – quick ontology mapping. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 683–697, Hiroshima (JP), 2004.

[Ehrig and Sure, 2004] Marc Ehrig and York Sure. Ontology mapping – an integrated approach. In *Proc. 1st European Semantic Web Symposium (ESWS)*, volume 3053 of *Lecture notes in computer science*, pages 76–91, Hersounisous (GR), May 2004.

[Ehrig and Sure, 2005] M. Ehrig and Y. Sure. Adaptive semantic integration. In *Proceedings of the ODBIS workshop at the 31st VLDB Conference*, Trondheim, Norway, September 2005.

[Ehrig *et al.*, 2003] M. Ehrig, P. Haase, F. van Harmelen, R. Siebes, S. Staab, H. Stuckenschmidt, R. Studer, and C. Tempich. The SWAP data and metadata model for semantics-based peer-to-peer systems. In *Proceedings of the First German Conference on Multiagent Technologies (MATES-2003)*, Lecture Notes in Artificial Intelligence. Springer, September 2003.

[Ehrig *et al.*, 2005] Marc Ehrig, Steffen Staab, and York Sure. Bootstrapping ontology alignment methods with APFEL. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 186–200, Galway (IE), 2005.

[Ehrig, 2006] Marc Ehrig. *Ontology alignment: bridging the semantic gap*. PhD thesis, Universität Fridericiana zu Karlsruhe, Karlsruhe (DE), 2006.

[Ehrig, 2007] Marc Ehrig. *Ontology alignment: bridging the semantic gap*. Semantic web and beyond: computing for human experience. Springer, New-York (NY US), 2007.

[Elfeky *et al.*, 2002] Mohamed Elfeky, Ahmed Elmagarmid, and Vassilios Verykios. Tailor: A record linkage tool box. In *Proc. 18th International Conference on Data Engineering (ICDE)*, pages 17–28, San Jose (CA US), 2002.

[Embley *et al.*, 2004] David Embley, Li Xu, and Yihong Ding. Automatic direct and indirect schema mapping: Experiences and lessons learned. *ACM SIGMOD Record*, 33(4):14–19, 2004.

[Euzenat and Shvaiko, 2007] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2007.

[Euzenat and Valtchev, 2003] Jérôme Euzenat and Petko Valtchev. An integrative proximity measure for ontology alignment. In *Proc. ISWC-2003 workshop on semantic information integration, Sanibel Island (FL US)*, pages 33–38, 2003.

[Euzenat and Valtchev, 2004] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proc. 16th European Conference on Artificial Intelligence (ECAI)*, pages 333–337, Valencia (ES), 2004.

[Euzenat *et al.*, 2003] Jérôme Euzenat, Nabil Layaïda, and Victor Dias. A semantic framework for multimedia document adaptation. In *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 31–36, Acapulco (MX), 2003.

[Euzenat *et al.*, 2004a] Jérôme Euzenat, Thanh Le Bach, Jesús Barrasa, Paolo Bouquet, Jan De Bo, Rose Dieng-Kuntz, Marc Ehrig, Manfred Hauswirth, Mustafa Jarrar, Rubén Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Ilya Zaihrayeu. State of the art on ontology alignment. Deliverable D2.2.3, Knowledge web NoE, 2004.

[Euzenat *et al.*, 2004b] Jérôme Euzenat, Marc Ehrig, and Raúl García Castro. Specification of a benchmarking methodology for alignment techniques. Deliverable D2.2.2, Knowledge web NoE, 2004.

[Euzenat *et al.*, 2005a] Jérôme Euzenat, Loredana Laera, Valentina Tamma, and Alexandre Viollet. Negotiation/argumentation techniques among agents complying to different ontologies. Deliverable 2.3.7, Knowledge web NoE, 2005.

[Euzenat *et al.*, 2005b] Jérôme Euzenat, François Scharffe, and Luciano Serafini. Specification of the delivery alignment format. Deliverable 2.2.6, Knowledge web NoE, 2005.

[Euzenat *et al.*, 2007] Jérôme Euzenat, Antoine Zimmermann, Marta Sabou, and Mathieu d'Aquin. Matching ontologies for context. deliverable 3.3.1, NeOn, 2007.

[Euzenat *et al.*, 2008] Jérôme Euzenat, Adrian Mocan, and François Scharffe. Ontology alignment: an ontology management perspective. In Martin Hepp, Pieter De Leenheer, Aldo De Moor, and York Sure, editors, *Ontology management: semantic web, semantic web services, and business applications*, chapter 6, pages 177–206. Springer, New-York (NY US), 2008.

[Euzenat, 1994] Jérôme Euzenat. Brief overview of T-tree: the Tropes taxonomy building tool. In *Proc. 4th ASIS SIG/CR Workshop on Classification Research*, pages 69–87, Columbus (OH US), 1994.

[Euzenat, 2000] Jérôme Euzenat. Towards formal knowledge intelligibility at the semiotic level. In *Proc. ECAI workshop on applied semiotics: control problems, Berlin (DE)*, pages 59–61, 2000.

[Euzenat, 2001] Jérôme Euzenat. Towards a principled approach to semantic interoperability. In *Proc. IJCAI Workshop on Ontologies and Information Sharing*, pages 19–25, Seattle (WA US), 2001.

[Euzenat, 2003] Jérôme Euzenat. Towards composing and benchmarking ontology alignments. In *Proc. ISWC Workshop on Semantic Integration*, pages 165–166, Sanibel Island (FL US), 2003.

[Euzenat, 2004] Jérôme Euzenat. An API for ontology alignment. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 698–712, Hiroshima (JP), 2004.

[Euzenat, 2005] Jérôme Euzenat. Alignment infrastructure for ontology mediation and other applications. In *Proc. International Workshop on Mediation in Semantic Web Services (MEDIATE)*, pages 81–95, Amsterdam (NL), 2005.

[Euzenat, 2007] Jérôme Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 248–253, Hyderabad (IN), 2007.

[Fagin *et al.*, 2003] Ronald Fagin, Phokion G. Kolaitis, R. J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proceedings of the 9th International Conference on Database Theory*, pages 207–224. Springer-Verlag, 2003.

[Falconer and Storey, 2007] Sean Falconer and Margaret-Anne Storey. A cognitive support framework for ontology mapping. In *Proc. 6th International Semantic Web Conference, Busan (KR)*, pages 114–127, 2007.

[Fellegi and Sunter, 1969] Ivan Fellegi and Alan Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[Fenton and Pfleeger, 1996] N. Fenton and L. Pfleeger, editors. *Software Metrics, A Rigorous & Practical Approach*. International Thomson Cmputer Press, 1996.

[Franconi and Tessaris, 2004] E. Franconi and S. Tessaris. Rules and queries with ontologies: a unified logical framework. In *Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR'04)*, 2004.

[Franconi *et al.*, 2001] Enrico Franconi, Ken Barker, and Diego Calvanese, editors. *Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII-2001)*. Springer-Verlag, 2001.

[Franconi *et al.*, 2003] Enrico Franconi, Gabriel Kuper, Andrei Lopatenko, and Luciano Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *Proc. VLDB International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, pages 64–76, Berlin (DE), 2003.

[Freksa, 1992] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227, 1992.

[Fung and Lo, 1998] P. Fung and Y.Y. Lo. An ir approach for translating new words from unparallel, comparable texts. In *Proc. of 36th Annual ACL Conference*, pages 414–420, Montreal, Canada, 1998.

[Frst and Trichet, 2005] F. Frst and F. Trichet. Axiom-based ontology matching. In *Proc. of the 3rd international conference on Knowledge capture (K-CAP'05)*, pages 195–196, New York, NY, USA, 2005. ACM Press.

[Gal *et al.*, 2004] Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, and Danilo Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB Journal*, 2004. to appear.

[Gale and Shapley, 1962] David Gale and Lloyd Stowell Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):5–15, 1962.

[Gandon, 2002] Fabien Gandon. *Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web*. Scientific philosopher doctorate thesis in informatics, INRIA and University of Nice - Sophia Antipolis, November 2002.

[Gangemi, 2005] Aldo Gangemi. Ontology design patterns for semantic web content. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 262–276, Hiroshima (JP), 2005.

[Ganter and Wille, 1999] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Verlag, Berlin (DE), 1999.

[Garbers *et al.*, 2006] J. Garbers, M. Niemann, and M. Mochol. A personalized hotel selection engine. In *Proc. of the Poster Session of 3rd ESWC 2006*, 2006.

[Ghidini and Giunchiglia, 2001] Chiara Ghidini and Fausto Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[Ghidini and Giunchiglia, 2004] Chiara Ghidini and Fausto Giunchiglia. A semantics for abstraction. In *Proc. ECAI*, 2004.

[Ghidini and Serafini, 1998] Chiara Ghidini and Luciano Serafini. Distributed first order logics. In *Proc. 2nd Conference on Frontiers of Combining Systems (FroCoS)*, pages 121–139, Amsterdam (NL), 1998.

[Ghidini and Serafini, 2006] Chiara Ghidini and Luciano Serafini. Reconciling concepts and relations in heterogeneous ontologies. In York Sure and John Domingue, editors, *Proc. of 3rd European Semantic Web Conference, Budva, Montenegro*, volume 4011 of *LNCS*, pages 50–64. Springer, 2006.

[Giunchiglia and Shvaiko, 2003a] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. In *Proc. IJCAI Workshop on Ontologies and Distributed Systems*, pages 139–146, Acapulco (MX), 2003.

[Giunchiglia and Shvaiko, 2003b] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2003.

[Giunchiglia and Walsh, 1992] Fausto Giunchiglia and Toby Walsh. A Theory of Abstraction. *Artificial Intelligence*, 57(2-3):323–390, 1992. Also IRST-Technical Report 9001-14, IRST, Trento, Italy.

[Giunchiglia and Yatskevich, 2004] Fausto Giunchiglia and Mikalai Yatskevich. Element level semantic matching. In *Proc. ISWC Meaning Coordination and Negotiation Workshop*, pages 37–48, Hiroshima (JP), 2004.

[Giunchiglia *et al.*, 2004] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proc. 1st European Semantic Web Symposium (ESWS)*, volume 3053 of *Lecture notes in computer science*, pages 61–75, Hersounisous (GR), 10-12 May 2004.

[Giunchiglia *et al.*, 2005] Fausto Giunchiglia, Mikalai Yatskevich, and Enrico Giunchiglia. Efficient semantic matching. In *Proc. 2nd European Semantic Web Conference (ESWC)*, volume 3532 of *Lecture notes in computer science*, pages 272–289, Hersounisous (GR), 2005.

[Giunchiglia, 1993] Fausto Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364, 1993. Short version in Proceedings IJCAI'93 Workshop on Using Knowledge in its Context, Chambery, France, 1993, pp. 39–49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.

[Goguen and Burstall, 1992] J. Goguen and R. Burstall. Institutions: abstract model theory for specification and programming. *Journal of the ACM*, 39, 1992.

[Goguen, 1991] J. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1:49–67, 1991.

[Goguen, 1999] Joseph Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphor, Analogy and Agents*, volume 1562 of *Lecture Notes in Artificial Intelligence*, pages 242–291. Springer Verlag, Berlin (DE), 1999.

[Goren-Bar and T.Kuflik, 2005] D. Goren-Bar and T.Kuflik. Supporting user-subjective categorization with self-organizing maps and learning vector quantization. *Journal of the American Society for Information Science and Technology JASIST*, 56(4):345–355, 2005.

[Grandzol, 2005] J. R. Grandzol. Improving the faculty selection process in highe education: A case for the analytic hierarchy process. *Using Advenced Tools, Tehciques, and Methodlogies. Association for Institutional Research*, 6, 2005.

[Grau *et al.*, 2004] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *Proceedings of the Third Internatonal Semantic Web Conference (ISWC2004)*, volume 3298 of *Lecture Notes in Computer Science*, 2004.

[Grau *et al.*, 2006] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Combining OWL ontologies using $\mathcal{E}$-connections. *Journal of web semantics*, 4(1):40–59, 2006.

[Gruber, 1995] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.

[Guarino, 1998] N. Guarino. Formal ontology and information systems. In *Proc. of the 1st International Conference on Formal Ontologies in Information Systems (FOIS'98)*, pages 3–15, 1998.

[Guttag and Horning, 1978] John V. Guttag and James J. Horning. The algebraic specification of abstract data types. *Acta informatica*, 10:27–52, 1978.

[Haas *et al.*, 2005] Laura Haas, Mauricio Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio grows up: from research prototype to industrial tool. In *Proc. 24th International Conference on Management of Data (SIGMOD)*, pages 805–810, Baltimore (MD US), 2005.

[Hahn, 2002] E. D. Hahn. Better decisions come from a results-based approach. *Marketing News*, 36(36):22–24, 2002.

[Hájek, 1998] Petr Hájek. *The metamathematics of fuzzy logic*. Kluwer, Dordrecht (NL), 1998.

[Halevy *et al.*, 2003] Alon Halevy, Zachary Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, 2003.

[Hamacher *et al.*, 1978] H. Hamacher, H. Leberling, and H.-J. Zimmermann. Sensitivity analysis in fuzzy linear programming. *Fuzzy Sets and Systems*, 1:269–281, 1978.

[Hameed *et al.*, 2004] Adil Hameed, Alun Preece, and Derek Sleeman. Ontology reconciliation. In Steffen Staab and Rudi Studer, editors, *Handbook on ontologies*, chapter 12, pages 231–250. Springer Verlag, Berlin (DE), 2004.

[Hayashi *et al.*, 1992] Y. Hayashi, E. Czogala, and J. J. Bukley. Fuzzy neural controller. In *Proc. IEEE Int. Conf. Fuzzy Syst*, pages 197–202, San Diago, 1992.

[He and Chang, 2006] Bin He and Kevin Chang. Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Transactions on Database Systems*, 31(1):1–45, 2006.

[He *et al.*, 2004] Hai He, Weiyi Meng, Clement Yu, and Zonghuan Wu. Automatic integration of web search interfaces with WISE-Integrator. *The VLDB Journal*, 13(3):256–273, 2004.

[He *et al.*, 2005] Hai He, Weiyi Meng, Clement Yu, and Zonghuan Wu. WISE-Integrator: A system for extracting and integrating complex web search interfaces of the deep web. In *Proc. 31st International Conference on Very Large Data Bases (VLDB)*, pages 1314–1317, Trondheim (NO), 2005.

[Heß and Kushmerick, 2004] Andreas Heß and Nicholas Kushmerick. Iterative ensemble classification for relational data: A case study of semantic web services. In *Proceedings of the 15th European Conference on Machine Learning*, Pisa, Italy, 2004.

[Hitzler *et al.*, 2005a] Pascal Hitzler, Jérôme Euzenat, Markus Krötzsch, Luciano Serafini, Heiner Stuckenschmidt, Holger Wache, and Antoine Zimmermann. Integrated view and comparison of alignment semantics. Deliverable 2.2.5, Knowledge web NoE, 2005.

[Hitzler *et al.*, 2005b] Pascal Hitzler, Markus Krötzsch, Marc Ehrig, and York Sure. What is ontology merging? – A category-theoretical perspective using pushouts. In Pavel Shvaiko, Jerome Euzenat, Alain Leger, Deborah L. McGuinness, and Holger Wache, editors, *Proceedings of the First International Workshop on Contexts and Ontologies: Theory, Practice and Applications (C&0). Workshop at the 20th National Conference on Artificial Intelligence, AAAI-05, Pittsburgh, Pennsylvania*, pages 104–107. AAAI Press, Menlo Park, California, July 2005. Technical Report WS-05-01.

[Horikawa *et al.*, 1992] S. Horikawa, T. Furuhashi, and Y. Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Trans. Neural Networks*, pages 801–806, 1992.

[Horrocks *et al.*, 2003] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[Horrocks *et al.*, 2004] Ian Horrocks, Peter Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: a semantic web rule language combining OWL and RuleML, 2004. http://www.w3.org/Submission/SWRL/.

[Hovy, 1998] Eduard Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proc. 1st International Conference on Language Resources and Evaluation (LREC)*, pages 535–542, Granada (ES), 1998.

[Hu *et al.*, 2005] Wei Hu, Ningsheng Jian, Yuzhong Qu, and Qanbing Wang. GMO: A graph matching for ontologies. In *Proc. K-CAP Workshop on Integrating Ontologies*, pages 43–50, Banff (CA), 2005.

[Humphrey, 1999] W. S. Humphrey, editor. *Introduction to the Team Software Process*. Addison-Wesley Professional, 1999.

[Huza *et al.*, 2006] Mirella Huza, Mounira Harzallah, and Francky Trichet. OntoMas: a tutoring system dedicated to ontology matching. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, pages 228–323, Athens (GA US), 2006.

[Ichise *et al.*, 2003] Ryutaro Ichise, Hideaki Takeda, and Shinichi Honiden. Integrating multiple internet directories by instance-based learning. In *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 22–30, Acapulco (MX), 2003.

[Ichise *et al.*, 2004] Ryutaro Ichise, Masahiro Hamasaki, and Hideaki Takeda. Discovering relationships among catalogs. In *Proc. 7th International Conference on Discovery Science*, volume 3245 of *Lecture notes in computer science*, pages 371–379, Padova (IT), 2004.

[Jacquemin and Royaute, 1994] Christian Jacquemin and Jean Royaute. Retrieving terms and their variants in a lexicalised unification-based framework. In *Proc. of 13th Annual International ACM-SIGIR Conference in Research and Development in Information Retrieval*, pages 132–141, Dublin, 1994.

[Jacquemin, 1996] Christian Jacquemin. What is the tree that we see through the window: A linguistic approach to windowing and term variation. *Information Processing and Management*, 32(4):445–458, 1996.

[Jacquemin, 1997] Christian Jacquemin. Recognition and acquisition: Two inter-related activities in corpus-based term extraction. *Terminology*, 4(2):245 ff., 1997.

[Jannink *et al.*, 1998] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. Encapsulation and composition of ontologies. In *Proceedings of the AAAI Workshop on AI & Information Integration*, 1998.

[Jarke *et al.*, 1999] M. Jarke, M. Lenzerini, Y. Vassilious, and P. Vassiliadis, editors. *Fundamentals of Data Warehousing*. Springer-Verlag, 1999.

[Jarke *et al.*, 2000] Mathias Jarke, V. Quix, D. Calvanese, Maurizio Lenzerini, Enrico Franconi, S. Ligoudistiano, P. Vassiliadis, and Yannis Vassiliou. Concept based design of data warehouses: The DWQ demonstrators. In *2000 ACM SIGMOD International Conference on Management of Data*, May 2000.

[Jean-Mary and Kabuka, 2007] Yves Jean-Mary and Mansur Kabuka. Asmov results for oaei 2007. In *Proc. 2nd Ontology matching workshop, Busan (KR)*, pages 141–150, 2007.

[Jian *et al.*, 2005] Ningsheng Jian, Wei Hu, Gong Cheng, and Yuzhong Qu. Falcon-AO: Aligning ontologies with Falcon. In *Proc. K-CAP Workshop on Integrating Ontologies*, pages 87–93, Banff (CA), 2005.

[Kalfoglou and Schorlemmer, 2002] Yannis Kalfoglou and Marco Schorlemmer. Information-flow-based ontology mapping. *Lecture notes in computer science*, 2519, 2002.

[Kalfoglou and Schorlemmer, 2003a] Yannis Kalfoglou and Marco Schorlemmer. IF-Map: an ontology mapping method based on information flow theory. *Journal on Data Semantics*, I:98–127, 2003.

[Kalfoglou and Schorlemmer, 2003b] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[Kalfoglou and Schorlemmer, 2005] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: The state of the art. In Yannis Kalfoglou, Marco Schorlemmer, Amit Sheth, Steffen Staab, and Michael Uschold, editors, *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005.

[Kang and Naughton, 2003] Jaewoo Kang and Jeffrey Naughton. On schema matching with opaque column names and data values. In *Proc. 22nd International Conference on Management of Data (SIGMOD)*, pages 205–216, San Diego (CA US), 2003.

[Keller and Hunt, 1992] J. M. Keller and D. J. Hunt. Evidence aggregation networks for fuzzy logic interface. *IEEE Trans. Neural Networks*, pages 751–769, 1992.

[Kensche *et al.*, 2005] David Kensche, Christoph Quix, Mohamed Amine Chatti, and Matthias Jarke. GeRoMe: A generic role based metamodel for model management. In *Proc. 4th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 3761 of *Lecture notes in computer science*, pages 1206–1224, Agia Napa (CY), 2005.

[Kent, 2000] R. Kent. The information flow foundation for conceptual knowledge organization. In *Proceedings of the Sixth International Conference of the International Society for Knowledge Organization*, 2000.

[Kerrigan *et al.*, 2007] Mike Kerrigan, Adrian Mocan, Martin Tanler, and Dieter Fensel. The web service modeling toolkit - an integrated development environment for semantic web services. In *Proc. 4th European Semantic Web Conference (ESWC) System Description Track*, pages 303–317, Innsbruck (AT), 2007.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[Kim *et al.*, 2005] Jaehong Kim, Minsu Jang, Young-Guk Ha, Joo-Chan Sohn, and Sang-Jo Lee. MoA: OWL ontology merging and alignment tool for the semantic web. In *Proc. 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and*

Expert Systems (IEA/AIE), volume 3533 of *Lecture notes in computer science*, pages 722–731, Bari (IT), 2005.

[Klein, 2001] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In *Proc. IJCAI Workshop on Ontologies and Information Sharing*, Seattle (WA US), 2001.

[Kotis and Vouros, 2004] Konstantinos Kotis and George Vouros. HCONE approach to ontology merging. In *Proc. 1st European Semantic Web Symposium (ESWS)*, volume 3053 of *Lecture notes in computer science*, pages 137–151, Hersounisous (GR), 2004.

[Kotis *et al.*, 2006] Konstantinos Kotis, George Vouros, and Konstantinos Stergiou. Towards automatic merging of domain ontologies: The HCONE-merge approach. *Journal of Web Semantics*, 4(1):60–79, 2006.

[Kutz *et al.*, 2004] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[Lacher and Groh, 2001] Martin Lacher and Georg Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proc. 14th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 305–309, Key West (FL US), 2001.

[Laera *et al.*, 2007] Loredana Laera, Ian Blacoe, Valentina Tamma, Terry Payne, Jérôme Euzenat, and Trevor Bench-Capon. Argumentation over ontology correspondences in MAS. In *Proc. 6th International conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1285–1292, Honolulu (HA US), 2007.

[Langlais *et al.*, 1998] Philippe Langlais, Jean Véronis, and Michel Simard. Methods and practical issues in evaluating alignment techniques. In *Proc. 17th International Conference on Computational Linguistics (CoLing)*, pages 711–717, Montréal (CA), 1998.

[Larson *et al.*, 1989] J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Trans. Softw. Eng.*, 15(4):449–463, 1989.

[Lausen *et al.*, 2005] Holger Lausen, Jos de Bruijn, Axel Polleres, , and Dieter Fensel. WSML – a language framework for semantic web services. In *Proceedings of the W3C Workshop on Rule Languages for Interoperability*, Washington (DC US), 2005.

[Le Berre, 2001] Daniel Le Berre. Jsat: The java satisfiability library, 2001.

[Lee *et al.*, 2002] Mong Li Lee, Liang Huai Yang, Wynne Hsu, and Xia Yang. XClust: clustering XML schemas for effective integration. In *Proc. 11th International Conference on Information and Knowledge Management (CIKM)*, pages 292–299, McLean (VA US), 2002.

[Léger *et al.*, 2005] Alain Léger, Lyndon Nixon, cois Paulus Fran Laurent Roquet, Malgorzata Mochol, Yannis Kompatsiaris, Vasileios Papastathis, Stamatia Drosopoulou, Mustafa Jarrar, Roberta Cuel, and Matteo Bonifacio. System and knowledge technology components for prototypical applications and business cases. Deliverable D1.1.4, Knowledge web NoE, 2005.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS-2002*, pages 233–246, 2002.

[Lerner, 2000] Barbara Staudt Lerner. A model for compound type changes encountered in schema evolution. *ACM Transactions on Database Systems*, 25(1):83–127, 2000.

[Levenshtein, 1966] I. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 1966.

[Li and Clifton, 1994] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proc. 20th International Conference on Very Large Data Bases (VLDB)*, pages 1–12, Santiago (CL), 1994.

[Li and Clifton, 2000] Wen-Syan Li and Chris Clifton. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering*, 33(1):49–84, 2000.

[Li *et al.*, 2005] L. Li, B. Wu, and Y. Yang. Agent-based Ontology Integration for Ontology-based Applications. In *Proc. of the Australasian Ontology Workshop (AOW 2005)*, volume 58, pages 53–59, 2005.

[Li *et al.*, 2006] Yi Li, Juanzi Li, Duo Zhang, and Jie Tang. Result of ontology alignment with RiMOM at OAEI-06. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, pages 181–190, Athens (GA US), 2006.

[Lim *et al.*, 1993] Ee-Peng Lim, Jaideep Srivastava, Satya Prabhakar, and James Richardson. Entity identification in database integration. In *Proc. 9th International Conference on Data Engineering (ICDE)*, pages 294–301, Wien (AT), 1993.

[Lin and Lu, 1995] C. T. Lin and Y. C. Lu. A neural fuzzy system with linguistic teaching signals. *IEEE Trans. Fuzzy Syst*, 3:169–189, 1995.

[Lin, 1998] Dekang Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conference of Machine Learning (ICML)*, pages 296–304, Madison (WI US), 1998.

[Lovins, 1968] Julie Beth Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(1):22–31, 1968.

[Lozano Tello and Gomez Perez, 2004] A. Lozano Tello and A. Gomez Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15:1–18, 2004.

[Mädche and Staab, 2002] Alexander Mädche and Steffen Staab. Measuring similarity between ontologies. In *Proc. 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, volume 2473 of *Lecture notes in computer science*, pages 251–263, Siguenza (ES), 2002.

[Mädche and Zacharias, 2002] Alexander Mädche and Valentin Zacharias. Clustering ontology-based metadata in the semantic web. In *Proc. 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 348–360, Helsinki (FI), 2002.

[Mädche *et al.*, 2002] Alexander Mädche, Boris Motik, Nuno Silva, and Raphael Volz. MAFRA – a mapping framework for distributed ontologies. In *Proc. 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, volume 2473 of *Lecture notes in computer science*, pages 235–250, Siguenza (ES), 2002.

[Madhavan *et al.*, 2001] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *Proc. 27th International Conference on Very Large Data Bases (VLDB)*, pages 48–58, Roma (IT), 2001.

[Madhavan *et al.*, 2002] Jayant Madhavan, Philip Bernstein, Pedro Domingos, and Alon Halevy. Representing and reasoning about mappings between domain models. In *Proc. 18th National Conference on Artificial Intelligence (AAAI)*, pages 122–133, Edmonton (CA), 2002.

[Madhavan *et al.*, 2005] Jayant Madhavan, Philip Bernstein, An-Hai Doan, and Alon Halevy. Corpus-based schema matching. In *Proc. 21st International Conference on Data Engineering (ICDE)*, pages 57–68, Tokyo (JP), 2005.

[Magnini *et al.*, 2003] B. Magnini, Luciano Serafini, and M. Speranza. Making explicit the semantics hidden in schema models. In *Proceedings of ISWC workshop on Human Language Technology for the Semantic Web and Web Services*, 2003.

[Mao and Peng, 2006] Ming Mao and Yefei Peng. PRIOR system: Results for OAEI 2006. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, pages 173–180, Athens (GA US), 2006.

[Mariño *et al.*, 1990] Olga Mariño, cois Rechenmann, Fran and Patrice Uvietta. Multiple perspectives and classification mechanism in object-oriented representation. In *9th European Conference on Artificial Intelligence*, pages 425–430, Stockholm, Suède, August 1990.

[Masolo *et al.*, 2003] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Ontology library. Deliverable D18, Wonderweb, 2003.

[Maßmann *et al.*, 2006] Sabine Maßmann, Daniel Engmann, and Erhard Rahm. COMA++: Results for the ontology alignment contest OAEI 2006. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, pages 107–114, Athens (GA US), 2006.

[Maynard and Ananiadou, 1999] D.G. Maynard and S. Ananiadou. Term extraction using a similarity-based approach. In *Recent Advances in Computational Terminology*. John Benjamins, 1999.

[Maynard, 1999] Diana Maynard. *Term Recognition Using Combined Knowledge Sources*. PhD thesis, Department of Computing and Mathematics, Manchester Metropolitan University, UK, 1999.

[McCray *et al.*, 1994] Alexa T. McCray, Suresh Srinivasan, and Allen C. Browne. Lexical methods for managing variation in biomedical terminologies. In *Proc. 18th Annual Symposium on Computer Applications in Medical Care (SCAMC)*, pages 235–239, Washington (US), 1994.

[McGuinness *et al.*, 2000] Deborah McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In *Proc. 7th International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 483–493, Breckenridge (CO US), 2000.

[MDC, 1999] Open information model, version 1.0. http://mdcinfo/oim/oim10.html, 1999.

[Meilicke *et al.*, 2006] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. Improving automatically created mappings using logical reasoning. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, pages 61–72, Athens (GA US), 2006.

[Meilicke *et al.*, 2007] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. Repairing ontology mappings. In *Proc. 22nd National conference on artificial intelligence (AAAI), Vancouver (CA)*, pages 1408–1413, 2007.

[Melnik *et al.*, 2002] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE)*, pages 117–128, San Jose (CA US), 2002.

[Melnik *et al.*, 2003a] Sergey Melnik, Erhard Rahm, and Philip Bernstein. Developing metadata-intensive applications with Rondo. *Journal of Web Semantics*, 1(1):47–74, 2003.

[Melnik *et al.*, 2003b] Sergey Melnik, Erhard Rahm, and Philip Bernstein. Rondo: A programming platform for model management. In *Proc. 22nd International Conference on Management of Data (SIGMOD)*, pages 193–204, San Diego (CA US), 2003.

[Melnik *et al.*, 2005] Sergey Melnik, Philip Bernstein, Alon Halevy, and Erhard Rahm. Supporting executable mappings in model management. In *Proc. 24th International Conference on Management of Data (SIGMOD)*, pages 167–178, Baltimore (MD US), 2005.

[Melnik, 2004] Sergey Melnik. *Generic Model Management Concepts and Algorithms*. Springer, Heidelberg (DE), 2004.

[Miles and Brickley, 2005a] Alistair Miles and Dan Brickley. SKOS core guide. Note, W3C, 2005.

[Miles and Brickley, 2005b] Alistair Miles and Dan Brickley. SKOS core vocabulary. Note, W3C, 2005.

[Miller *et al.*, 2000] Renée Miller, Laura Haas, and Mauricio Hernández. Schema mapping as query discovery. In *Proc. 26th International Conference on Very Large Data Bases (VLDB)*, pages 77–88, Cairo (EG), 2000.

[Miller, 1995] George Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Milo and Zohar, 1998] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proc. 24th International Conference on Very Large Data Bases (VLDB)*, pages 122–133, New York (NY US), 1998.

[Mitra and Wiederhold, 2002] P. Mitra and Gio Wiederhold. Resolving terminological heterogeneity in ontologies. In *Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence (ECAI)*, 2002.

[Mitra *et al.*, 1999] Prasenjit Mitra, Gio Wiederhold, and Jan Jannink. Semi-automatic integration of knowledge sources. In *Proc. 2nd International Conference on Information Fusion*, pages 572–581, Sunnyvale (CA US), 1999.

[Mitra *et al.*, 2005] Prasenjit Mitra, Natalya Noy, and Anuj Jaiswal. Ontology mapping discovery with uncertainty. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 537–547, Galway (IE), 2005.

[Mocan and Ciampian, 2005] Adrian Mocan and Emilia Ciampian. Mappings creation using a view based approach. In *Proc. of the 1st International Workshop on Mediation in Semantic Web Services (Mediate)*, pages 97–112, Amsterdam (NL), 2005.

[Mocan and Cimpian, 2007] Adrian Mocan and Emilia Cimpian. An ontology-based data mediation framework for semantic environments. *International journal on semantic web and information systems*, 3(2):66–95, 2007.

[Mocan *et al.*, 2006] Adrian Mocan, Emilia Cimpian, and Mick Kerrigan. Formal model for ontology mapping creation. In *Proc. 5th International Semantic Web Conference (ISWC)*, volume 4273 of *Lecture notes in computer science*, pages 459–472, Athens (GA US), 2006.

[Mochol and Paslaru Bontas Simperl, 2006] Malgorzata Mochol and Elena Paslaru Bontas Simperl. Practical guidelines for building semantic erecruitment applications. In *Proc. of the International Conference on Knowledge Management (iKnow'06), Special Track: Advanced Semantic Technologies*, 2006.

[Modica *et al.*, 2001] Giovanni Modica, Avigdor Gal, and Hasan Jamil. The use of machine-generated ontologies in dynamic information seeking. In *Proc. 9th International Conference on Cooperative Information Systems (CoopIS)*, volume 2172 of *Lecture notes in computer science*, pages 433–448, Trento (IT), 2001.

[Munkres, 1957] James Munkres. Algorithms for the assignment and transportation problems. *SIAM Journal on Applied Mathematics*, 5(1):32–38, 1957.

[Navathe and Buneman, 1986] S. Navathe and P. Buneman. Integrating user views in database design. *Computer*, 19(1):50–62, January 1986.

[Ngai *et al.*, 2002] Grace Ngai, Marine Carpuat, and Pascale Fung. Identifying concepts across languages: A first step towards a corpus-based approach to automatic ontology alignment. In *Proc. of COLING 2002*, Taipei, Taiwan, 2002.

[Niles and Pease, 2001] I. Niles and A. Pease. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, *In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.

[Nottelmann and Straccia, 2005] Henrik Nottelmann and Umberto Straccia. sPLMap: A probabilistic approach to schema matching. In *Proc. 27th European Conference on Information Retrieval Research (ECIR)*, pages 81–95, Santiago de Compostela (ES), 2005.

[Noy and Musen, 2000] Natalya Noy and Mark Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th National Conference on Artificial Intelligence (AAAI)*, pages 450–455, Austin (TX US), 2000.

[Noy and Musen, 2001] Natalya Noy and Mark Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Proc. IJCAI Workshop on Ontologies and Information Sharing*, pages 63–70, Seattle (WA US), 2001.

[Noy and Musen, 2002a] Natalya Noy and Mark Musen. Evaluating ontology-mapping tools: requirements and experience. In *Proc. 1st EKAW Workshop on Evaluation of Ontology Tools (EON)*, pages 1–14, Siguenza (ES), 2002.

[Noy and Musen, 2002b] Natalya Noy and Mark Musen. PromptDiff: A fixed-point algorithm for comparing ontology versions. In *Proc. 18th National Conference on Artificial Intelligence (AAAI)*, pages 744–750, Edmonton (CA), 2002.

[Noy and Musen, 2003] Natalya Noy and Marc Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

[Noy, 2004] Natalya Noy. Tools for mapping and merging ontologies. In Steffen Staab and Rudi Studer, editors, *Handbook on ontologies*, chapter 18, pages 365–384. Springer Verlag, Berlin (DE), 2004.

[Oberle *et al.*, 2004] Daniel Oberle, Raphael Volz, Steffen Staab, and Boris Motik. An extensible ontology software environment. In Steffen Staab and Rudi Studer, editors, *Handbook on ontologies*, chapter 15, pages 299–319. Springer Verlag, Berlin (DE), 2004.

[Palma and Haase, 2005] Raúl Palma and Peter Haase. Oyster: Sharing and re-using ontologies in a peer-to-peer community. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 1059–1062, Galway (IE), 2005.

[Palopoli *et al.*, 2000] L. Palopoli, G. Terracina, and D. Ursino. The system dike: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *Proceeding of ADBIS-DASFAA*, pages 108–117, 2000.

[Palopoli *et al.*, 2003] Luigi Palopoli, Giorgio Terracina, and Domenico Ursino. DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases. *Software–Practice and Experience*, 33(9):847–884, 2003.

[Pan *et al.*, 2005] Rong Pan, Zhongli Ding, Yang Yu, and Yun Peng. A Bayesian network approach to ontology mapping. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 563–577, Hiroshima (JP), 2005.

[Pao, 1989] Y.H. Pao. *Adaptive Pattern Recognition and Neural networks*. Addison-Wesley., 1989.

[Papadimitriou and Steiglitz, 1998] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, 1998.

[Parent and Spaccapietra, 2000] Christine Parent and Stefano Spaccapietra. Database integration: the key to data interoperability. In Mike Papazoglou, Stefano Spaccapietra, and Zahir Tari, editors, *Object-oriented data modeling*, chapter 9, pages 221–253. The MIT Press, Cambridge (MA US), 2000.

[Paslaru Bontas and Mochol, 2005] Elena Paslaru Bontas and Malgorzata Mochol. Towards a reuse-oriented methodology for ontology engineering. In *Proc. of 7th International Conference on Terminology and Knowledge Engineering (TKE 2005)*, 2005.

[Paslaru Bontas *et al.*, 2005] Elena Paslaru Bontas, Malgorzata Mochol, and Robert Tolksdorf. Case Studies on Ontology Reuse. In *Proc. of the 5th International Conference on Knowledge Management*, 2005.

[Patel *et al.*, 2003] C. Patel, K. Supekar, and Y. Lee. Ontogenie: Extracting ontology instances from WWW. In *Human Language Technology for the Semantic Web and Web Services, ISWC'03*, Sanibel Island, Florida, 2003.

[Pedersen *et al.*, 2004] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet::similarity - measuring the relatedness of concepts. In *Appears in the Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004.

[Peim *et al.*, 2002] Martin Peim, Enrico Franconi, Norman Paton, and Carole Goble. Query processing with description logic ontologies over object-wrapped databases. In *Proc. of the 14th International Conference on Scientific and Statistical Database Management (SSDBM'02)*, July 2002.

[Peim *et al.*, 2004] Martin Peim, Enrico Franconi, and Norman Paton. Applying functional languages in knowledge-based information integration systems. In Peter Gray, Larry Kerschberg, Peter King, and Alex Poulovassilis, editors, *The Functional Approach to Data Management*. Springer-Verlag, 2004.

[Pierce, 1991] Benjamin Pierce. *Basic Category Theory for Computer Scientists*. Foundations of Computing. The MIT press, Cambridge, MA, 1991.

[Porter, 1980] Martin Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Qu *et al.*, 2006] Yuzhong Qu, Wei Hu, and Gong Chen. Constructing virtual documents for ontology matching. In *Proc. 15th International World Wide Web Conference (WWW)*, pages 23–31, Edinburgh (UK), 2006.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[Rauch and Šimůnek, 2005] Jan Rauch and Milan Šimůnek. An alternative approach to mining association rules. In T. Y. Lin, S. Ohsuga, C. J. Liau, and S. Tsumoto, editors, *Data Mining: Foundations, Methods, and Applications*, pages 211–232. Springer, 2005.

[Rector *et al.*, 2004] Alan L. Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In *Proc. 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, volume 3257 of *Lecture notes in computer science*, pages 63–81, Whittlebury Hall (UK), 2004.

[Reiter, 1992] Raymond Reiter. What should a database know? *Journal of Logic Programming*, 14(2,3), 1992.

[Resnik, 1995] Phillip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. 14th IJCAI*, pages 448–453, Montréal, Canada, 1995.

[Resnik, 1999] Phillip Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[Roman *et al.*, 2004] Dumitru Roman, Holger Lausen, and Uwe Keller. Web service modeling ontology standard (WSMO-standard). Working Draft D2v0.2, WSMO, 2004.

[Saatly, 1990] T. L. Saatly. How to Make a Decision: The Analytic Hierarchy Process. *European Journal of Operational Research*, 48(1):9–26, 1990.

[Sager, 1990] J. C. Sager. *A Practical Course in Terminology Processing*. John Benjamins, 1990.

[Salton, 1989] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[Sayyadian *et al.*, 2005] Mayssam Sayyadian, Yoonkyong Lee, An-Hai Doan, and Arnon Rosenthal. Tuning schema matching software using synthetic scenarios. In *Proc. 31st International Conference on Very Large Data Bases (VLDB)*, pages 994–1005, Trondheim (NO), 2005.

[Scharffe and de Bruijn, 2005] François Scharffe and Jos de Bruijn. A language to specify mappings between ontologies. In *Proc. IEEE Conference on Internet-Based Systems (SITIS)*, Yaounde (CM), 2005.

[Scharffe and Kiryakov, 2005] François Scharffe and Atanas Kiryakov. Omwg d7.2: Mapping and merging tool design. Working Draft D7.2v0.2, OMWG, 2005.

[Scharffe, 2005] François Scharffe. Mapping and merging tool design. Deliverable D7.2, Ontology Management Working Group, 2005.

[Scharffe, 2007] Franois Scharffe. Dynamerge: A merging algorithm for structured data integration on the web. In *Proc. DASFAA 2007 International Workshop on Scalable Web Information Integration and Service (SWIIS)*, Bangkok (TH), 2007.

[Schorlemmer *et al.*, 2002] M. Schorlemmer, S. Potter, and D. Robertson. Automated support for composition of transformtional components in knowledge engineering. Technical Report EDI-INF-RR-0137, Division of Informatics, University of Edinburgh, 2002.

[Sebastiani, 2002] F. Sebastiani. Machine learning in automated text categorization. *ACM Comuting Surveys*, 34(1):1–47, 2002.

[Serafini and Ghidini, 2000] Luciano Serafini and Chiara Ghidini. Using wrapper agents to answer queries in distributed information systems. In *Proceedings of the First Biennial Int. Conf. on Advances in Information Systems (ADVIS-2000)*, 2000.

[Serafini and Tamilin, 2005] Luciano Serafini and Andrei Tamilin. DRAGO: Distributed reasoning architecture for the semantic web. In *Proc. 2nd European Semantic Web Conference (ESWC)*, volume 3532 of *Lecture notes in computer science*, pages 361–376, Hersounisous (GR), May 2005.

[Serafini *et al.*, 2005] Luciano Serafini, Alex Borgida, and Andrei Tamilin. Aspects of distributed and modular ontology reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI-05*, Edinburgh, Scotland, 2005.

[Sheth *et al.*, 1988] Amit P. Sheth, James A. Larson, Aloysius Cornelio, and Shamkant B. Navathe. A tool for integrating conceptual schemas and user views. In *Proceedings of the Fourth International Conference on Data Engineering*, pages 176–183. IEEE Computer Society, 1988.

[Shvaiko and Euzenat, 2005] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, IV:146–171, 2005.

[Shvaiko *et al.*, 2006] Pavel Shvaiko, Jérôme Euzenat, Natalya Noy, Heiner Stuckenschmidt, Richard Benjamins, and Michael Uschold, editors. *Proc. 1st ISWC International Workshop on Ontology Matching (OM)*, Athens (GA US), 2006.

[Shvaiko *et al.*, 2007] Pavel Shvaiko, Jérôme Euzenat, Heiner Stuckenschmidt, Malgorzata Mochol, Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, Willem Robert van Hage, Ondrej Svab, and Vojtech Svatek. Description of alignment evaluation and benchmarking results. deliverable 2.2.9, Knowledge web NoE, 2007.

[Shvaiko, 2004a] Pavel Shvaiko. A classification of schema-based matching approaches. Technical Report DIT-04-09, University of Trento, `http://eprints.biblio.unitn.it/archive/00000654/01/093.pdf`, December 2004.

[Shvaiko, 2004b] Pavel Shvaiko. Iterative schema-based semantic matching. Technical Report DIT-04-020, University of Trento (IT), 2004.

[Shvaiko, 2006] Pavel Shvaiko. *Iterative Schema-based Semantic Matching*. PhD thesis, International Doctorate School in Information and Communication Technology, University of Trento, Trento (IT), November 2006.

[Silva and Rocha, 2003] Nuno Silva and J. Rocha. An ontology mapping framework for the semantic web. In *Proc. 6th International Conference on Business Information Systems*, Colorado Springs, USA, 2003.

[Sirin *et al.*, 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: a practical OWL-DL reasoner. *Journal of Web Semantics*, 5, 2007. To appear.

[Sowa, 1984] John Sowa. *Conceptual Structures: Information Processing In Mind and Machine*. Addison-Wesley, 1984.

[Staab and Studer, 2004] Steffen Staab and Rudi Studer. *Handbook on ontologies*. International handbooks on information systems. Springer Verlag, Berlin (DE), 2004.

[Stefani *et al.*, 2003] F. Stefani, D. Macii, A. Moschitta, and D. Petri. Fft benchmarking for digital signal processing technologies. In *17th IMEKO World Congress*, Dubrovnik, Croatia, 22-27 June 2003.

[Stoilos *et al.*, 2005] Georgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A string metric for ontology alignment. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 624–637, Galway (IE), 2005.

[Straccia and Troncy, 2005a] Umberto Straccia and Raphaël Troncy. oMAP: An implemented framework for automatically aligning owl ontologies. In *Proceedings of the* 2^{nd} *Italian Semantic Web Workshop (SWAP'05)*, Trento, Italy, 2005.

[Straccia and Troncy, 2005b] Umberto Straccia and Raphaël Troncy. oMAP: Combining classifiers for aligning automatically OWL ontologies. In *Proc. 6th International Conference on Web Information Systems Engineering (WISE)*, pages 133–147, New York (NY US), 2005.

[Straccia and Troncy, 2005c] Umberto Straccia and Raphaël Troncy. oMAP: Results of the ontology alignment contest. In *Proceedings of the K-Cap 2005 Workshop on Integrating Ontologies*, pages 92–96, 2005.

[Straccia and Troncy, 2006] Umberto Straccia and Raphaël Troncy. Towards distributed information retrieval in the semantic web: Query reformulation using the oMAP framework. In *Proc. 3rd European Semantic Web Conference (ESWC)*, volume 4011 of *Lecture notes in computer science*, pages 378–392, Budva (ME), 2006.

[Stuckenschmidt *et al.*, 2005] Heiner Stuckenschmidt, Marc Ehrig, Jérôme Euzenat, Andreas Hess, Robert van Hage, Wei Hu, Ningsheng Jian, Gong Chen, Yuzhong Qu, George Stoilos, Giorgo Stamou, Umberto Straccia, Vojtech Svatek, Raphaël Troncy, Petko Valtchev, and Mikalai Yatskevich. Description of alignment implementation and benchmarking results. deliverable 2.2.4, Knowledge web NoE, 2005.

[Stumme and Mädche, 2001] Gerd Stumme and Alexander Mädche. FCA-Merge: Bottom-up merging of ontologies. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 225–234, Seattle (WA US), 2001.

[Su and Gulla, 2003] X. Su and J. A. Gulla. Semantic enrichment for ontology mapping. In *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems*, Manchester, UK, 2003.

[Sun and Lin, 2001] Aixin Sun and Ee-Peng Lin. Hierarchical text classification and evaluation. In *Proc. 1st International Conference on Data Mining (ICDM)*, pages 521–528, San Jose (CA), 2001.

[Sure *et al.*, 2004] York Sure, Oscar Corcho, Jérôme Euzenat, and Todd Hughes, editors. *Proc. 3rd ISWC Workshop on Evaluation of Ontology-based tools (EON)*, Hiroshima (JP), 2004.

[Svab *et al.*, 2005] O. Svab, V. Svatek, P. Berka, D. Rak, and P. Tomasek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Proceedings of the 5th International Semantic Web Conference ISWC-05*, 2005. Poster Track.

[Tatarinov and Halevy, 2004] Igor Tatarinov and Alon Halevy. Efficient query reformulation in peer data management systems. In *Proceedings of the SIGMOD International Conference on Management of Data (SIGMOD'04)*, 2004.

[Tounazi, 2004] Mohamed Tounazi. Alignement d'ontologies dans OWL. Master's thesis, University of Montréal, 2004.

[Tu and Yu, 2005] Kewei Tu and Yong Yu. CMC: Combining multiple schema-matching strategies based on credibility prediction. In *Proc. 10th International Conference on Database Systems for Advanced Applications (DASFAA)*, volume 3453 of *Lecture notes in computer science*, pages 888–893, Beijing (CN), 2005.

[Tverski, 1977] Amos Tverski. Features of similarity. *Psychological Review*, 84(2):327–352, 1977.

[Uehara and Fujise, 1992] K. Uehara and M. Fujise. Multistage fuzzy inference formulated as linguistic-truth-value propagation and its learning algorithm based on back-propagating error information. *IEEE Transactions on Fuzzy Systems*, 3, 1992.

[Ullman, 1997] Jeffrey Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf on Database Theory (ICDT'97)*, pages 19–40, 1997.

[Uschold and Jasper, 1999] M. Uschold and R. Jasper. A Framework for Understanding and Classifying Ontology Applications, 1999.

[Uschold, 2005] Mike Uschold. Achieving semantic interoperability using RDF and OWL - v4, 2005. http://lists.w3.org/Archives/Public/public-swbp-wg/2005Sep/att-0027/SemanticII-v4.htm.

[Valtchev and Euzenat, 1997] Petko Valtchev and Jérôme Euzenat. Dissimilarity measure for collections of objects and values. In *Proc. 2nd Symposium on Intelligent Data Analysis (IDA)*, volume 1280 of *Lecture notes in computer science*, pages 259–272, London (UK), 1997.

[Valtchev, 1999] Petko Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, Grenoble (FR), 1999.

[van Rijsbergen, 1975] Cornelis Joost (Keith) van Rijsbergen. *Information retrieval*. Butterworths, London (UK), 1975. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[Velegrakis *et al.*, 2003] Yannis Velegrakis, Renée Miller, and Lucian Popa. Mapping adaptation under evolving schemas. In *Proc. 29th International Conference on Very Large Data Bases (VLDB)*, pages 584–595, Berlin (DE), 2003.

[Velegrakis *et al.*, 2004] Yannis Velegrakis, Renée Miller, and Lucian Popa. Preserving mapping consistency under schema changes. *The VLDB Journal*, 13(3):274–293, 2004.

[Verheijen *et al.*, 1998] Danladi Verheijen, Gio Wiederhold, Jan Jannink, and Srinivasan Pichai. Encapsulation and composition of ontologies. In *The Fifteenth National Conference on Artificial Intelligence, AAAI'98, Madison, Wisconsin (USA)*, July 1998.

[Visser *et al.*, 2002] Ubbo Visser, Thomas Vögele, and Christoph Schlieder. Spatio-terminological information retrieval using the buster system. In *Proceedings of the EnviroInfo*, pages 93–100, Wien (AT), 2002.

[Vögele *et al.*, 2003] Thomas Vögele, Sebastian Hübner, and Gerhard Schuster. Buster - an information broker for the semantic web. *Künstliche Intelligenz*, 3:31–34, July 2003.

[Vouros and Kotis, 2005] George Vouros and Konstantinos Kotis. Extending HCONE-merge by approximating the intended interpretations of concepts iteratively. In *Proc. 2nd European Semantic Web Conference (ESWC)*, volume 3532 of *Lecture notes in computer science*, pages 198–210, Hersounisous (GR), May 2005.

[Šváb *et al.*, 2007] Ondřej Šváb, Vojtěch Svátek, and Heiner Stuckenschmidt. A study in empirical and 'casuistic' analysis of ontology mapping results. In *Proc. 4th European Semantic Web Conference (ESWC)*, Innsbruck (AU), 2007.

[Wang *et al.*, 2004] Jiying Wang, Ji-Rong Wen, Frederick Lochovsky, and Wei-Ying Ma. Instance-based schema matching for web databases by domain-specific query probing. In *Proc. 30th International Conference on Very Large Data Bases (VLDB)*, pages 408–419, Toronto (CA), 2004.

[Waterfeld *et al.*, 2008] Walter Waterfeld, Moritz Weiten, and Peter Haase. Ontology management infrastrutures. In Martin Hepp, Pieter De Leenheer, Aldo De Moor, and York Sure, editors, *Ontology management: semantic web, semantic web services, and business applications*, chapter 3, pages 39–89. Springer, New-York (NY US), 2008.

[Winkler, 1999] William Winkler. The state of record linkage and current research problems. Technical Report 99/04, Statistics of Income Division, Internal Revenue Service Publication, 1999.

[Xu and Embley, 2003] Li Xu and David Embley. Discovering direct and indirect matches for schema elements. In *Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 39–46, Kyoto (JP), 2003.

[Yager, 1992] Ronald Yager. OWA neurons: a new class of fuzzy neurons. In *Proc. Int. Joint Conf. Neural Networks*, volume 1, pages 226–231, 1992.

[Zhang and Bodenreider, 2007a] Songmao Zhang and Olivier Bodenreider. Experience in aligning anatomical ontologies. *International Journal on Semantic Web and Information Systems*, 3(2), 2007.

[Zhang and Bodenreider, 2007b] Songmao Zhang and Olivier Bodenreider. Reconciling concepts and relations in heterogeneous ontologies. In *Proc. 12th International Health (Medical) Informatics Congress (Medinfo)*, Brisbane (AUS), 2007. to appear.

[Zhang *et al.*, 2004] Songmao Zhang, Peter Mork, and Olivier Bodenreider. Lessons learned from aligning two representations of anatomy. In *Proc. 13th International Conference on the Principles of Knowledge Representation and Reasoning Conference (KR)*, pages 555–560, Whistler (CA), 2004.

[Zhang *et al.*, 2005] Songmao Zhang, Peter Mork, Olivier Bodenreider, and Philip Bernstein. Comparing two approaches for aligning representations of anatomy. *Artificial Intelligence in Medicine*, 2005. Submitted.

[Zhdanova and Shvaiko, 2006] Anna Zhdanova and Pavel Shvaiko. Community-driven ontology matching. In *Proc. 3rd European Semantic Web Conference (ESWC)*, volume 4011 of *Lecture notes in computer science*, pages 34–49, Budva (ME), 2006.

[Zimmermann and Euzenat, 2006] Antoine Zimmermann and Jérôme Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *Proc. 5th International Semantic Web Conference (ISWC)*, volume 4273 of *Lecture notes in computer science*, pages 16–29, Athens (GA US), 2006.

Related deliverables

A number of Knowledge web deliverables are clearly related to this one:

Project	Number	**Title** and relationship
KW	D1.1.4	**System and knowledge technology components for prototypical applications and business cases** introduces the use case that has been more specifically considered here.
KW	D1.2.4	**Architecture of the semantic web framework** describes the semantic web framework and in particular the components for ontology matching that this deliverable is supposed to help choosing.
KW	D2.1.2.2v1	**Report on realizing practical approximate and distributed reasoning for ontologies** elaborates on D2.1.1 and extends the study of modularity for the purpose of scalability.
KW	D2.1.3.1	**Report on modularization of ontologies** elaborates on D2.1.1 and studies partitioning algorithms for large ontologies into smaller modules, distributed reasoning, engineering approaches to ontology modularization, and composition, i.e. the inverse to modularization.
KW	D2.1.1	**Survey of scalability techniques for reasoning with ontologies** provided an in-depth discussion about benchmarking techniques that have been mentioned here.
KW	D2.1.4	**Specification of a methodology, general criteria, and test suites for benchmarking ontology tools** provided a general framework for defining a benchmarking test.
KW	D2.4.2	**Heterogeneity in the context of Semantic web services - use cases and usage scenarios** provides a more detailed description of use cases in the context of semantic web services.
SDK	D19.2	**WSMO in Knowledge web** describes the involvement work package in the wider WSMO context.
SDK	D13.3	**Mediation** describes the approach to mediation in the Web Service Execution Environment. This is a detailled test case for the use of alignment techniques in the context of Web services.
SEKT	D4.4.1	**Ontology Mediation Management V1** defines the Mapping language at the source of the OMWG-ML described here.
OMWG	D7.2	**Ontology Mapping Language RDF/XML Syntax** defines the RDF/XML syntax for the language presented here after the work we have done for this deliverable.