

Semantic web semantics

(M2R SWXO lecture notes)

Jérôme Euzenat
INRIA & LIG,
Montbonnot, France
`Jerome.Euzenat@inria.fr`

February 7, 2012

Copyright © Jérôme Euzenat, 2007-2011
This document can be distributed as such or used in part with attribution.

Chapter 1

Introduction

The web has been constantly evolving from a distributed hypertext system to a very large information processing machine. As fast as it is, this evolution is grounded on theoretical principles borrowing to several fields of computer science such as programming languages, data bases, structured documentation, logic and artificial intelligence. The smooth operation of the past and future web at a large scale is relying on these foundations.

These lecture notes introduce the semantics of knowledge representation on the web (semantic web technologies). Further, they aim at acquainting students with the techniques of knowledge representation so that they can use them in other contexts. In particular, we aim at providing:

- a deep understanding of these technologies, at least deeper than the simple language approach;
- a concrete application example of logics, out of logics itself.

The semantic web extends the web with richer and more precise information because it is expressed in a formal language using a vocabulary defined in an ontology (a structured vocabulary of concepts and properties defined in a logic). Ontologies are used for describing Web resource content and reasoning about these resources formally. We introduce the semantic Web languages (RDF, RDFS, OWL) and show their relations with knowledge representation formalisms (conceptual graphs, description logics) and XML. This provides the tools for reasoning with ontologies and, in particular, to evaluate queries. However, the distributed nature of the web leads to heterogeneous ontologies which must be matched before using them. We show how to match ontologies and how to semantically interpret the relations between ontologies. Finally, this is applied to network of peers using knowledge together.

The material is presented so as to be as generic as possible, however it is illustrated through specific and practical semantic web technologies (mostly language).

1.1 Lecture note format and topic dependencies

This document constitutes the lecture notes of a master course (see §1.2) that has been taught entirely or partially between 2007 and 2011 in various formats at the University of Grenoble following my previous lectures on Semantics of knowledge representation [Euzenat, 2002] (in French).

In its entirety, the course is made of 7 sessions of 3h.

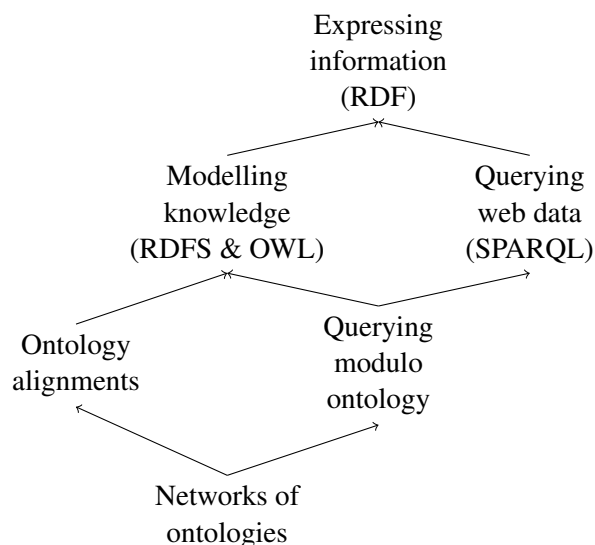


Figure 1.1: Dependencies between the various topics of these lectures.

3h00 Expressing information (RDF)

6h00 Modelling knowledge (RDFS and OWL)

3h00 Querying web data (SPARQL)

3h00 SPARQL Extensions

3h00 Ontology alignments

3h00 Networks of ontologies

It is further divided in three parts: Expressing knowledge (data and ontologies), Querying and Networks of ontologies.

The dependencies between the main parts are provided in Figure 1.1. However, the course may also be given or read in a reduced form by considering only Chapter 2, 3, 4, 5, 8, 9, 11, and 13.1.

1.2 Context in the semantic web course

This section provides a connection with the part of the course dedicated to the foundations of XML taught by Pierre Genevès and Nabil Layaïda.

They have considered expressing information within the XML language. XML is not a very expressive language: it provides a structure which stores information and this information has to be extracted in order to be accessed and exploited. It does not allow for expressing knowledge in terms of descriptions which will be used for guiding this interpretation.

XML provides a database point of view about knowledge representation in which the power of what can be done with the language depends on the power of the query language (Xpath, Xquery, etc.) and the representation is a store.

In the context of the semantic web, this language is provided with a semantics specifying how to interpret the structure. This allows for defining general statements which will apply to the relevant data whatever its syntactic form.

Doing so requires the definition of a logic, specifying the semantics of statements (or formulas) and identifying the possible models of a document or a set of documents. Answering queries will then have to be performed through inference sanctioned by their correctness and completeness with regard to the semantics.

There is anyway a continuity between these manipulation of the data on the web. It starts with extracting data from documents through navigation:

$$\text{Document (XML)} \vdash \text{Query(xpath)}$$

The information can also be extracted modulo a document schema. Although schemas are here in the first place for specifying acceptable document structures, they often provide shortcuts and can help producing more guarantees on query answering:

$$\text{Document (XML)} + \text{Schema (DTD, XML Schema)} \vdash \text{Query (xpath)}$$

In addition, sophisticated query languages provide the possibility to reason about queries through their algebraic structures and the opportunity to arrange extracted information in a better way:

$$\text{Document (XML)} + \text{Schema (XML Schema)} \vdash \text{Query (xquery)}$$

Semantic web languages, as presented here, manipulate data roughly in the same way as this is done in XML, hence:

$$\text{Data (RDF)} \models \phi (\text{RDF})$$

however, we use \models to emphasise that the operations for extracting information are defined semantically. Similarly to XQuery, the SPARQL query language has been defined for querying RDF data:

$$\text{Data (RDF)} \vdash \text{Query (SPARQL)}$$

Although, defined syntactically in the SPARQL recommendation, it is preferable to define SPARQL semantically:

$$\text{Data (RDF)} \models \text{Query (SPARQL)}$$

and to show the completeness of the syntactic description with respect to the RDF semantics. In addition, semantic web technologies allow for expressing ontologies which are theories constraining how data must be interpreted. RDF and its extension RDFS (RDF Schema) together with OWL form the three formal logics recommended by W3C for representing knowledge on the semantic web. In this presentation, the only operation which will be considered is the test that data modulo some ontologies entails a particular formula:

$$\text{Data (RDF)} + \text{Ontology (RDF Schema, OWL)} \models \phi (\text{RDF})$$

We will concentrate on the entailment test and consider further elaborate query languages later. Finally, the semantic web inherits the distributed aspect of the web. This means that independent data and ontology sources provide information which has to be interpreted together. For that purpose, one may consider a set Ω of ontologies and data sources (still described in OWL and RDF) and a set of alignments Λ which expresses relations between entities in the ontologies. Hence, extracting information from the semantic web, is obtained within such a network of ontologies, through:

Data (RDF) + Ontology (RDF Schema, OWL) $\models_{\Omega, \Lambda} \phi$ (RDF)

The frontiers between the “syntactic web” and the semantic web tends to vanish as XML gain expressiveness (with XML Schema and Xquery) and RDF allows for minimal reasoning. The definition of XML and query languages, such as XPath, in logics, such as the μ -calculus, calls for a closer understanding of both worlds. This is one of the main reasons of putting these lectures together. Developing semantic web applications, like in many other cases, is characterised by a trade-off between the expressivity of the languages used and the complexity of reasoning.

So, expressing knowledge and information on the web depends on different languages:

- data language (XML, RDF);
- ontology language (XML Schema, RDF Schema, OWL)
- query language (XPath, Xquery, SPARQL)
- distributed aspect (Alignments)

each one needing to be semantically defined.

1.3 Exercises

We offer exercises which are usually based on former exams. They have been dispatched at the end of chapters so as to be considered as soon as the chapter is covered. The exams (90mn) were respectively made of:

- Exercises 1, 4, 8 (and additionally 14);
- Exercises 2, 5, 6, 9, and 11;
- Exercises 3, 7, 10, 12, and 13.

1.4 Further information

The web site of the course is <http://exmo.inrialpes.fr/teaching/sw/>. It contains references to other material as well as information on the first part of the course.

Any comment, issue or clarification enquiry may be send by email to the author:

Jerome.Euzenat@inria.fr

Further books covering parts of these topics are [Antoniou and van Harmelen, 2008; Hitzler *et al.*, 2009].

1.5 Acknowledgements

Most of the material presented here comes from the collective effort for establishing W3C recommendations.

Other parts of these notes have been extracted from part of my own works some of them non published. Finally, important parts of Chapter 5 and 6.1 come from my student Faisal Alkhateeb’s thesis [Alkhateeb, 2008] and parts of Chapter 11 and 12 are results from Antoine Zimmermann’s PhD dissertation [Zimmermann, 2008].

Thanks to the students of these courses for their questions and criticisms.

Contents

1	Introduction	3
1.1	Lecture note format and topic dependencies	3
1.2	Context in the semantic web course	4
1.3	Exercises	6
1.4	Further information	6
1.5	Acknowledgements	6
I	Graphs and ontologies	9
2	RDF: Resource description framework	11
2.1	URI (and XML): the basic layer	11
2.2	RDF Syntax	12
2.3	Simple RDF Semantics	14
2.4	Inference mechanism	18
2.5	Computational properties	19
2.6	Conclusion	20
2.7	Exercises	20
3	Ontology languages	23
3.1	RDF Schema	23
3.2	The web ontology language OWL	31
3.3	Conclusion	42
3.4	Exercises	42
II	Queries	45
4	Querying the semantic web	47
4.1	Motivation	47
4.2	What is a query language?	48
5	Querying RDF with SPARQL	51
5.1	Syntax	51
5.2	SPARQL Semantics	54
5.3	Algebraic manipulation	56
5.4	Entailment regimes	57

5.5 Exercises	57
6 Extending SPARQL	61
6.1 PSPARQL	61
7 Querying modulo ontologies	65
7.1 RDF(S) closure and query answering	66
7.2 The nSPARQL approach	67
7.3 Answering SPARQL queries modulo RDFS through PSPARQL	70
7.4 Querying modulo DL-lite	71
7.5 Conclusion	72
7.6 Exercises	72
III Networks of ontologies	73
8 Networks of ontologies and alignments	75
8.1 Motivation	75
8.2 Reminder: ontology semantics	77
9 Alignment syntax and networks of ontologies	79
9.1 Networks of ontologies	83
10 Alignment algebra	87
10.1 Relation algebra	88
10.2 Aggregating matcher results	89
10.3 Composing alignments	91
10.4 Algebraic reasoning with alignments	92
10.5 Algebra granularity	94
11 Alignment semantics	97
11.1 Consistence, consequence and closure	99
11.2 Local models from the standpoint of an ontology	102
11.3 Conclusion	103
11.4 Exercises	104
12 Equalising semantics for alignments	105
13 Application: Distributed query evaluation	109
13.1 Semantic peer-to-peer systems	109
13.2 The semantics of semantic peer-to-peer systems	110
13.3 Exercises	111
13.4 Conclusions	113
14 Conclusion	115
IV Correction of exercises	117

Part I

Graphs and ontologies

Chapter 2

RDF: Resource description framework

The Resource Description Framework (RDF) is a W3C recommended language for expressing data on the semantic web [Manola and Miller, 2004]. The atomic constructs of RDF are *statements*, which are triples (subject, predicate, object) consisting of the resource (the subject) being described, a property (the predicate), and a property value (the object).

This section is devoted to the presentation of *Simple RDF*, i.e., RDF without specific (RDF or RDFS) vocabulary [Brickley and Guha, 2004]. We first recall (Section 2.2) its abstract syntax [Carroll and Klyne, 2004], its semantics (Section 2.3, using the notions of simple interpretations, models, simple entailment of [Hayes, 2004]), then Section 2.4 uses homomorphisms to characterize simple RDF entailment (as done in [Baget, 2005] for a graph-theoretic encoding of RDF, and in [Gutierrez *et al.*, 2004] for a database encoding), instead of the equivalent interpolation lemma of [Hayes, 2004]. Section 3.1.4 introduces the RDF entailment problem and its complexity.

2.1 URI (and XML): the basic layer

A URI (uniform resource identifier [Berners-Lee *et al.*, 1998]) generalizes URL (uniform resource locator) for identifying not only web pages but any resource (human, book, an author property). An *uriref* is a URI with a fragment, e.g., `http://www.example.org/homepage.html#section1`. A URI is meant to denote one fixed resource, but a resource may have several URIs.

The semantic web relies loosely on XML. XML is only considered as a transport format.

RDF is rather defined as an abstract object. A collection of RDF statements (RDF triples) can be intuitively understood as a directed labelled graph: resources are nodes and statements are arcs (from the subject node to the object node) connecting the nodes.

The language is provided with a model-theoretic semantics [Hayes, 2004], that defines the notion of consequence (or entailment) between two RDF graphs, i.e., when an RDF graph is entailed by another one.

Answers to an RDF query (the knowledge base and the query are RDF graphs) are determined by the consequence, and can be computed using a particular *map* (a mapping from terms of the query to terms of the knowledge base preserving constants), a *graph homomorphism* [Gutierrez *et al.*, 2004; Baget, 2005].

OWL, in the last OWL 2 recommendation [Beckett, 2009a], specifies five different syntaxes:

XML/RDF is the only mandatory exchange syntax, it is based on XML with a very regular, but verbose, structure;

OWL/XML an ugly but straightforward XML syntax;

Functional syntax as its name indicates, this syntax expresses the abstract trees of the OWL concepts and is better suited for specifying its semantics (and manipulating it inductively);

Manchester syntax is supposedly a “user-friendly” syntax, using syntactic sugar;

Triple itself an extension of the n3 syntax for RDF, it is also supposed to be more readable by users.

It is clear that, in spite of their names, the import of these languages is not their syntax, but their structure and semantics.

2.2 RDF Syntax

RDF can be expressed in a variety of formats including RDF/XML [Beckett, 2009b], Turtle [Beckett, 2006], or n3. We use here its abstract syntax (triple format), which is sufficient for most purposes. To define the syntax of RDF, we need to introduce the *terminology* over which RDF graphs are constructed.

RDF terminology

Definition 1 (RDF terminology [Hayes, 2004]). *The RDF terminology \mathcal{T} is the union of three pairwise disjoint infinite sets of terms :*

- the set \mathcal{U} of urirefs,
- the set \mathcal{L} of literals (itself partitioned into two sets, the set \mathcal{L}_p of plain literals and the set \mathcal{L}_t of typed literals), and
- the set \mathcal{B} of variables.

The set $\mathcal{V} = \mathcal{U} \cup \mathcal{L}$ of names is called the vocabulary.

From now on, we use different notations for the elements of these sets: a variable will be prefixed by ? (like ?b1), a literal will be between quotation marks (like "27"), and the rest will be urirefs (like foaf:Person — foaf:¹ is a name space prefix used for representing personal information — ex:friend or simply friend).

2.2.1 RDF graphs as triples

RDF graphs are constructed over sets of URI references (or urirefs), blanks, and literals [Carroll and Klyne, 2004]. Because we want to stress the compatibility of the RDF structure with classical logic, we will use the term variable instead of that of “blank” which is a vocabulary specific to RDF. The specificity of blanks with regard to variables is their quantification. Indeed, a blank in RDF is an existentially quantified variable over a particular graph. So, we prefer to retain this classical interpretation which is useful when an RDF graph is put in a different context.

¹<http://xmlns.com/foaf/spec/>

Definition 2 (RDF graph). An RDF triple is an element of $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times \mathcal{T}$. An RDF graph is a finite set of RDF triples.

Excluding variables as predicates and literals as subject was an unnecessary restriction in the RDF design, that has been relaxed in many RDF extensions. These constraints complexify the syntax specification, and relaxing them neither changes RDF semantics nor the computational properties of reasoning. In consequence, we adopt such an extension introduced in [ter Horst, 2005] and called *generalized RDF graphs*, or simply GRDF graphs.

Definition 3 (GRDF graph). A GRDF triple is an element of $\mathcal{T} \times (\mathcal{U} \cup \mathcal{B}) \times \mathcal{T}$. A GRDF graph is a finite set of GRDF triples.

So, every RDF graph is a GRDF graph.

Example 1. The following set of triples represents a GRDF graph:

$$\left\{ \begin{array}{l} \langle ?b1 \quad foaf:name \quad "Faisal" \rangle, \\ \langle ?b1 \quad ex:daughter \quad ?b2 \rangle, \\ \langle ?b2 \quad ?b4 \quad ?b3 \rangle, \\ \langle ?b3 \quad foaf:knows \quad ?b1 \rangle, \\ \langle ?b3 \quad foaf:name \quad ?name \rangle \end{array} \right\}$$

*Intuitively, this graph means that there exists an entity named (*foaf:name*) "Faisal" that has a daughter (*ex:daughter*) that has some relation with another entity whose name is non determined, and that knows (*foaf:knows*) the entity named "Faisal".*

Notations If G is an RDF graph, we use $\mathcal{T}(G)$, $\mathcal{U}(G)$, $\mathcal{L}(G)$, $\mathcal{B}(G)$, $\mathcal{V}(G)$ to denote the set of terms, urirefs, literals, variables or names that appear in at least one triple of G . In a triple $\langle s, p, o \rangle$, s is called the subject, p the predicate and o the object. We denote by $subj(G)$ the set $\{s \mid \langle s, p, o \rangle \in G\}$ of elements appearing as a subject in a triple of a GRDF graph G . $pred(G)$ and $obj(G)$ are defined in the same way for predicates and objects. We call $nodes(G)$ the *nodes* of G , the set of elements appearing either as subject or object in a triple of G , i.e., $subj(G) \cup obj(G)$. A *term* of G is an element of $term(G) = subj(G) \cup pred(G) \cup obj(G)$. If $\mathcal{Y} \subseteq \mathcal{T}$ is a set of terms, we denote $\mathcal{Y} \cap term(G)$ by $\mathcal{Y}(G)$. For instance, $\mathcal{V}(G)$ is the set of names appearing in G .

A *ground* GRDF graph G is a GRDF graph with no variables, i.e., $term(G) \subseteq \mathcal{V}$.

2.2.2 Graph representation of RDF triples

A *simple GRDF graph* can be represented graphically as a directed labeled multigraph $\langle N, E, \gamma, \lambda \rangle$ where the set of nodes N is the set of terms appearing as a subject or object in at least one triple of G , the set of arcs E is the set of triples of G , γ associates to each arc a pair of nodes (its extremities) $\gamma(e) = \langle \gamma_1(e), \gamma_2(e) \rangle$ where $\gamma_1(e)$ is the source of the arc e and $\gamma_2(e)$ its target; finally, λ labels the nodes and the arcs of the graph: if s is a node of N , i.e., a term, then $\lambda(s) = s$, and if e is an arc of E , i.e., a triple (s, p, o) , then $\lambda(e) = p$, i.e., if $\langle s, p, o \rangle$ is a triple, then $s \xrightarrow{p} o$ is an arc. When drawing such graphs, the nodes resulting from literals are represented by rectangles while the others are represented by rectangles with rounded corners.

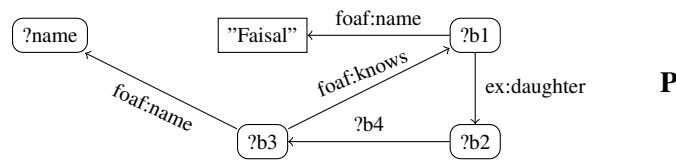


Figure 2.1: A GRDF graph.

For example, the GRDF triples given in Example 1 can be represented graphically as in Figure 2.1.

In what follows, we do not distinguish between the two views of the RDF syntax (as sets of triples or directed labeled graphs). We will then speak interchangeably about their nodes, their arcs, or the triples which make them up.

2.3 Simple RDF Semantics

[Hayes, 2004] introduces several semantics for RDF graphs. In this section, we present only the *simple semantics* without RDF/RDFS vocabulary [Brickley and Guha, 2004]. The definitions of interpretations, models, satisfiability, and entailment correspond to the *simple interpretations*, *simple models*, *simple satisfiability*, and *simple entailments* of [Hayes, 2004]. RDF and RDFS consequences (or entailments) can be polynomially reduced to simple entailment via RDF or RDFS rules [Baget, 2005; ter Horst, 2005] (see Section 3.1.3).

2.3.1 Interpretations

An interpretation describes possible way(s) the world might be in order to determine the truth-value of any ground RDF graph. It does this by specifying for each uriref, what is its denotation? In addition, if it is used to indicate a property, what values that property has for each thing in the universe?

Interpretations that assign particular meanings to some names in a given vocabulary will be named from that vocabulary, e.g., RDFS interpretations (see Section 3.1). An interpretation with no particular extra conditions on a vocabulary (including the RDF vocabulary itself) will be simply called an interpretation.

Definition 4 (Interpretation of a vocabulary). *Let $V \subseteq \mathcal{V} = \mathcal{U} \cup \mathcal{L}$ be a vocabulary, an interpretation of V is a tuple $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ such that:*

- I_R is a set of resources that contains $V \cap \mathcal{L}$;
- $I_P \subseteq I_R$ is a set of properties;
- $I_{EXT} : I_P \rightarrow 2^{I_R \times I_R}$ associates to each property a set of pairs of resources called the extension of the property;
- the interpretation function $\iota : V \rightarrow I_R$ associates to each name in V a resource of I_R , if $v \in \mathcal{L}$, then $\iota(v) = v$.

2.3.2 Models

By providing RDF with formal semantics, [Hayes, 2004] expresses the conditions under which an interpretation is a model for an RDF graph. The usual notions of validity, satisfiability and

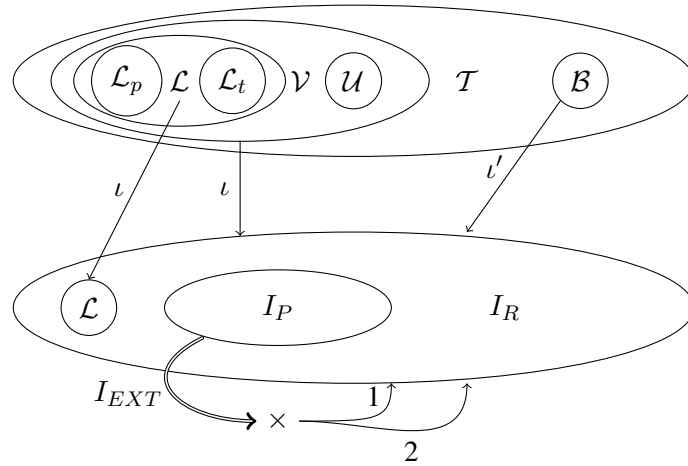


Figure 2.2: Domain structure for Simple RDF semantics.

consequence are entirely determined by these conditions.

Intuitively, a ground triple $\langle s, p, o \rangle$ in a GRDF graph will be true under the interpretation I if p is interpreted as a property (for example, r_p), s and o are interpreted as resources (for example, r_s and r_o , respectively), and the pair of resources $\langle r_s, r_o \rangle$ belongs to the extension of the property r_p . A triple $\langle s, p, ?b \rangle$ with the variable $?b \in \mathcal{B}$ would be true under I if there exists a resource r_b such that the pair $\langle r_s, r_b \rangle$ belongs to the extension r_p . When interpreting a variable node, an arbitrary resource can be chosen. To ensure that a variable always is interpreted by the same resource, extensions of the interpretation function is defined as follow.

Definition 5 (Extension to variables). *Let $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ be an interpretation of a vocabulary $V \subseteq \mathcal{V}$, and $B \subseteq \mathcal{B}$ a set of variables. An extension of ι to B is a mapping $\iota' : \mathcal{V} \cup B \rightarrow I_R$ such that $\forall x \in V, \iota'(x) = \iota(x)$.*

An interpretation I is a model of GRDF graph G if all triples are true under I .

Definition 6 (Model of a GRDF graph). *Let $V \subseteq \mathcal{V}$ be a vocabulary, and G be a GRDF graph such that every name appearing in G is also in V ($\mathcal{V}(G) \subseteq V$). An interpretation $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ of V is a model of G iff there exists an extension ι' that extends ι to $\mathcal{B}(G)$ such that for each triple $\langle s, p, o \rangle$ of G , $\iota'(p) \in I_P$ and $\langle \iota'(s), \iota'(o) \rangle \in I_{EXT}(\iota'(p))$. The mapping ι' is called a proof of G in I .*

2.3.3 Satisfiability, validity, entailment and consequence

The following definition is the standard model-theoretic definition of satisfiability validity and consequence.

Definition 7 (Satisfiability, validity, consequence). *A graph G is satisfiable iff it has a model. G is valid iff for every interpretation I of a vocabulary $V \supseteq \mathcal{V}(G)$, I is a model of G . A graph G' is a consequence of a graph G , or G entails G' , denoted $G \models_{GRDF} G'$, iff every model of G is also a model of G' .*

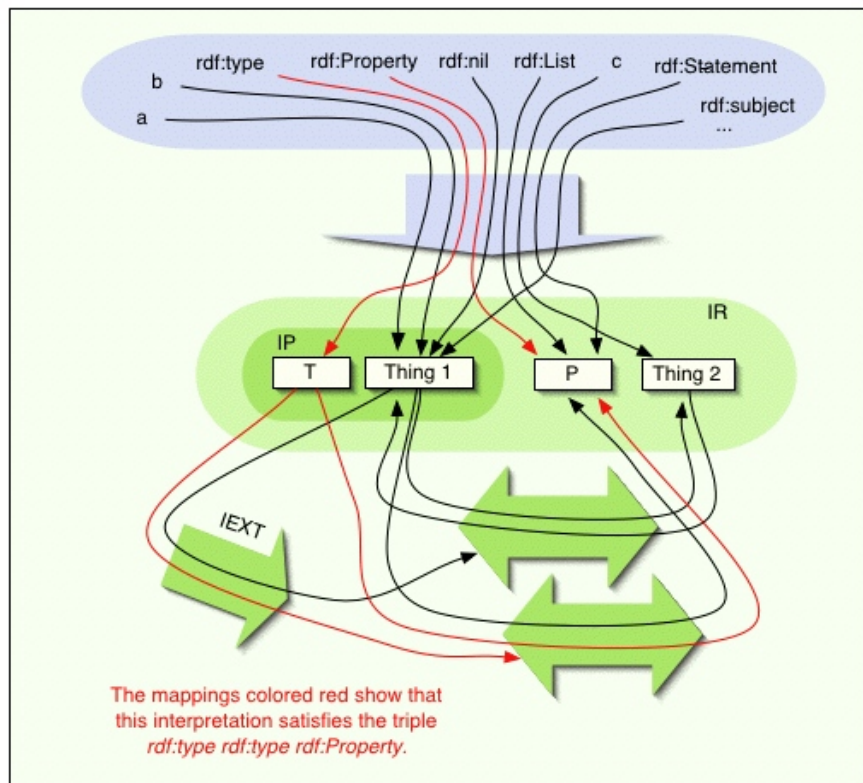


Figure 2.3: Example of interpretation of the domain structure (from [Hayes, 2004]).

Proposition 1 (Empty graph, subgraph, instance, merging lemmata [Hayes, 2004]). *Here are simple properties given in [Hayes, 2004]:*

Empty graph lemma *the empty set of triple is entailed by any graph and does not entail any graph but itself;*

Subgraph lemma *a graph entails all its subgraphs (i.e., subsets of triples);*

Instance lemma *a graph is entailed by any of its instances (i.e., variables substituted by values);*

Merging lemma *A set of graphs entails the union of the graphs which entails any of these graphs.*

Sketch of proof. The subgraph lemma holds because any model of a subgraph imposes less constraints on interpretations than the graph it is a subgraph of. Hence, any of its interpretation will be a interpretation of the supergraph. This also entails the Empty lemma, because the empty graph is a subgraph of any graph. Concerning the Instance lemma, the instantiated graph imposes more constraints than the initial graph, because the assignment of the corresponding variable is fixed. Hence, any model of the instance is a model of the general graph (with the extension to blank assigning the instantiated value to each instantiated variable). Finally, any graph is a subgraph of its union with other graphs which, by the Subgraph lemma, makes it entailed. \square

Currently the two following propositions rely on the same construct: the Herbrand model or isomorphic model.

Proposition 2 (Satisfiability, validity [Baget, 2005; ter Horst, 2005]). *Every GRDF graph is satisfiable. The only valid GRDF graph is the empty graph.*

Proof. (Satisfiability) See proof of Proposition 3.

(Validity) a non empty GRDF graph has no proof in an interpretation in which all properties are interpreted by I_{EXT} as an empty set. \square

Proposition 3 (Interpolation lemma [Hayes, 2004]).

$$G \models_{GRDF} H \text{ iff a subgraph of } G \text{ is an instance of } H$$

Proof. \Leftarrow : This is clearly true due to the subgraph and instance lemmas: if G' is a subgraph of G , then $G \models G'$ because all models of G trivially satisfy the triples of G' (subgraph lemma). If G' is an instance of H , then $G' \models H$ because, for each model $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ of G' , there exist an extension to variables ι' satisfying all triples and for each triple $\langle s', p', o' \rangle \in G'$ instantiating a triple $\langle s, p, o \rangle \in H$, there is an extension of ι'' to variables, such that for each element x in the triple, $\iota''(x) = \iota'(x')$ if $x \in \mathcal{B}(H)$ *mathcal* G' , and $\iota''(x) = \iota'(x)$ otherwise (instance lemma).

\Rightarrow : The Herbrand interpretation \hat{I}^G of the graph G is defined by $\hat{I}^G = \langle \hat{I}_R^G, \hat{I}_P^G, \hat{I}_{EXT}^G, \hat{\iota}^G \rangle$ such that:

- $\hat{I}_R^G = \text{term}(G)$;
- $\hat{I}_P^G = \text{pred}(G)$;
- $\hat{I}_{EXT}^G : \hat{I}_P^G \rightarrow 2^{\hat{I}_R^G \times \hat{I}_R^G}$ such that $\forall p \in \hat{I}_P^G, \hat{I}_{EXT}^G(p) = \{ \langle s, o \rangle \in \hat{I}_R^G \times \hat{I}_R^G; \langle s, p, o \rangle \in G \}$;

- the interpretation function $\hat{\iota}^G : V \rightarrow \hat{I}_R^G$ is the identity function over $term(G)$.

\hat{I}^G is clearly a model of G , since it can be extended by the identity function on blanks and trivially satisfies all constraints ($\forall x \in term(G), \iota'(x) = \iota(x) = x$). The condition of Definition 6 immediately follows from this construction. This proves the satisfiability part of Proposition 2.

If $G \models_{GRDF} H$, then \hat{I}^G is also a model of H . This means that there exists an extension $\hat{\iota}^H$ of $\hat{\iota}^G$ to blanks satisfying every triple in H . Hence, there must exist a map $\hat{\iota}^H : \mathcal{B}(H) \cup \mathcal{V}(G) \rightarrow \hat{I}_R^G$, such that for any triple $\langle s, p, o \rangle \in H$, $\langle \hat{\iota}^H(s), \hat{\iota}^H(o) \rangle \in \hat{I}_{EXT}^G(\hat{\iota}^H(p))$. But given the definition of \hat{I}_{EXT}^G , this means that $\langle \hat{\iota}^H(s), \hat{\iota}^H(p), \hat{\iota}^H(o) \rangle \in G$. For each element x of this triple, either $x \in \mathcal{V}(G)$ (and then $\hat{\iota}^H(x) = x$) or $x \in \mathcal{B}(H)$ (and then $\hat{\iota}^H(x) \in term(G)$). This triple is thus necessarily an instance of $\langle s, p, o \rangle$. So, $\forall \langle s, p, o \rangle \in H$, there exists an instantiation $\langle s', p', o' \rangle \in G$, of it, which proves that a subgraph of G is an instantiation of H . \square

2.4 Inference mechanism

SIMPLE RDF ENTAILMENT [Hayes, 2004] can be characterized as a kind of graph homomorphism. A *graph homomorphism* from an RDF graph H into an RDF graph G , as defined in [Baget, 2005; Gutierrez *et al.*, 2004], is a mapping π from the nodes of H into the nodes of G preserving the arc structure, i.e., for each node $x \in H$, if $\lambda(x) \in \mathcal{U} \cup \mathcal{L}$ then $\lambda(\pi(x)) = \lambda(x)$; and each arc $x \xrightarrow{p} y$ is mapped to $\pi(x) \xrightarrow{\pi(p)} \pi(y)$. This definition is similar to the projection used to characterize entailment of conceptual graphs [Chein and Mugnier, 2009] (see [Corby *et al.*, 2000] for precise relationship between RDF and conceptual graphs). We modify this definition to the one that maps $term(H)$ into $term(G)$. Maps are used to ensure that a variable always mapped to the same term, as done for extensions to interpretations.

Definition 8 (Map). *Let $V_1 \subseteq \mathcal{T}$, and $V_2 \subseteq \mathcal{T}$ be two sets of terms. A map from V_1 to V_2 is a mapping $\mu : V_1 \rightarrow V_2$ such that $\forall x \in (V_1 \cap \mathcal{V}), \mu(x) = x$.*

The *map* defined in [Gutierrez *et al.*, 2004; Pérez *et al.*, 2006] is a particular case of Definition 8. An RDF homomorphism is a map preserving the arc structure.

Definition 9 (GRDF homomorphism). *Let G and H be two GRDF graphs. A GRDF homomorphism from H into G is a map π from $term(H)$ to $term(G)$ such that $\forall \langle s, p, o \rangle \in H$, $\langle \pi(s), \pi(p), \pi(o) \rangle \in G$.*

[Gutierrez *et al.*, 2004] provides without proof an equivalence theorem (Theorem 3) between RDF entailment and maps. A proof is provided in [Baget, 2005] also for RDF graphs, but the homomorphism involved is a mapping from nodes to nodes, and not from terms to terms. In RDF, the two definitions are equivalent. However, the terms-to-terms version is necessary to extend the theorem of RDF (Theorem 4) to the PRDF graphs studied in [Alkhateeb *et al.*, 2009]. All the proofs are available in [Alkhateeb, 2008].

Example 2 (GRDF homomorphism). *Figure 2.4 shows two GRDF graphs Q and G (note that the graph Q is the graph P of Figure 2.1, to which the following triple is added $\langle ?b3, foaf:mbox, ?mbox \rangle$. The map π_1 defined by $\{("Faisal", "Faisal"), (?b1, ?c1), (?name, "Natasha"), (?b2, ?c2), (?b4, ex:friend), (?mbox, "natasha@yahoo.com"), (?b3, ex:Person1)\}$ is a GRDF homomorphism from Q into G . And the map π_2 defined by $\{("Faisal", "Faisal"), (?b1, ?c1), (?name, "Deema"), (?b3, ex:Person2), (?b4,$*

$(\text{ex:friend}), (?b2, ?c2)\}$ is a GRDF homomorphism from P into G . Note that π_2 cannot be extended to a GRDF homomorphism from Q into G since there is no mailbox for "Deema" in G .

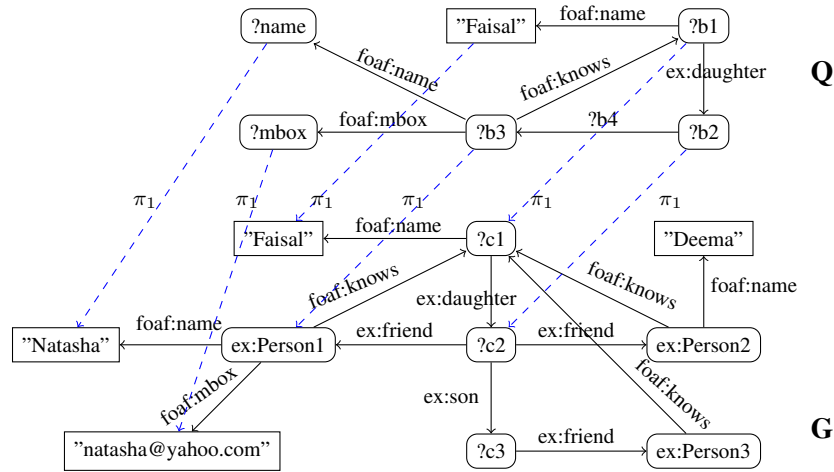


Figure 2.4: A GRDF homomorphism from Q into G .

Theorem 4. Let G and H be two GRDF graphs, then $G \models_{GRDF} H$ if and only if there is a GRDF homomorphism from H into G .

Proof. The proof of this theorem is left as an exercise (it is given in [Alkhateeb, 2008]). It relies on showing that an homomorphism from H into G determines a subgraph of G (the image of H in G by the homomorphism) and that this subgraph is necessarily an instance of H by Definition 9. Then, by Proposition 3, the equivalence is shown. \square

This equivalence between the semantic notion of entailment and the syntactic notion of homomorphism is the ground by which a correct and complete query answering procedure can be designed. More precisely, the set of answers to a GRDF graph query Q over an RDF knowledge base G are the set of RDF homomorphisms from Q into G which, by Theorem 4, correspond to RDF consequence.

2.5 Computational properties

The decision problem associated to simple RDF semantics is called SIMPLE RDF ENTAILMENT, and is defined as follows:

SIMPLE (G)RDF ENTAILMENT

Instance: two GRDF graphs G and H .

Question: Does $G \models_{GRDF} H$?

SIMPLE (G)RDF ENTAILMENT is an NP-complete problem for RDF graphs [Gutierrez *et al.*, 2004]. For GRDF graphs, its complexity remains unchanged [Pérez *et al.*, 2006]. Polynomial subclasses of the problem have been exhibited based upon the structure or labeling of the query:

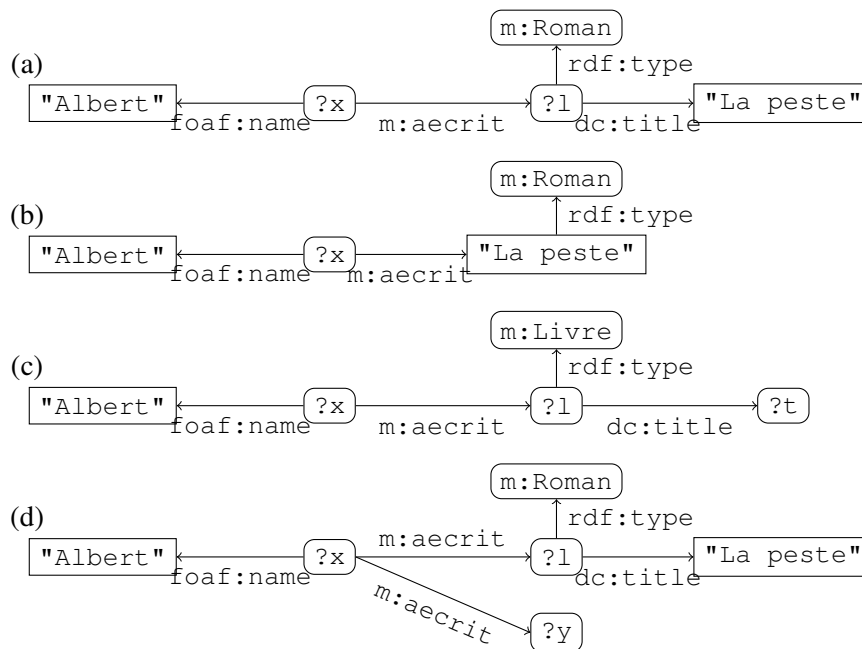


Figure 2.5: RDF graphs.

- when the query is ground [ter Horst, 2004], or more generally if it has a bounded number of variables,
- when the query is a tree or admits a bounded decompositions into a tree, according to the methods in [Gottlob *et al.*, 1999] as shown in [Baget, 2005].

2.6 Conclusion

RDF (or Simple RDF) is a language allowing for expressing information as a graph in which resources are related by predicates. This language has a model theoretic semantics allowing for defining the notion of consequence (or entailment) between such graphs. Moreover, entailment is equivalent to the existence of an homomorphism between the entailed graph and the entailing graph. In its generality, checking for entailment is an NP-complete problem.

This allows to check if a formula (query) is a consequence of the information expressed in an RDF graph.

2.7 Exercises

Exercise 1 (RDF assertions). Consider the four graphs of Figure 2.5.

1. Are they all well-formed RDF graphs? Why?
2. Express the graph of Figure 2.5(a) as a set of triples. You will list the sets of literals, URIRefs and variables (or blanks) found in this graph.
3. Give an informal meaning of this graph or its expression in the predicate calculus

4. Consider the RDF graphs of Figure 2.5 which are well-formed, do some of them entail others? Explain why.

Exercise 2 (RDF and assertions). Consider the two graphs of Figure 2.6 dealing with the expression of social relationships.

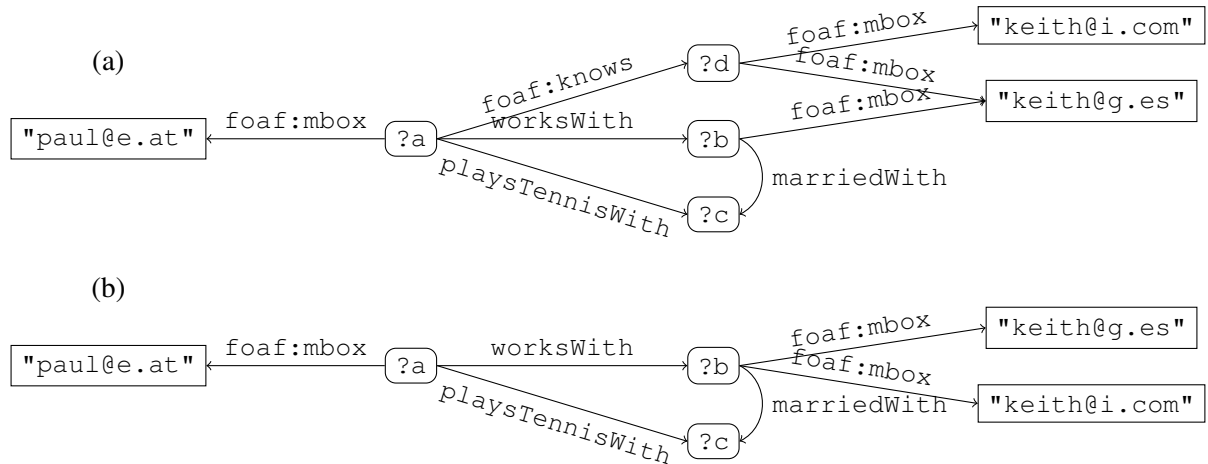


Figure 2.6: RDF graphs.

1. Express the graph of Figure 2.6(b) as a set of triples. You will give the list of literals, URIRefs and variables (or blanks) in this graph.
2. What is, informally, the meaning of the graph of Figure 2.6(b) (tell it in English or French or predicate calculus)?
3. Does one of these graphs entail the other? (explain why)

Exercise 3 (RDF graphs). Here are the 8 triples of an RDF graph G about writers and their works: (all identifiers correspond in fact to URIs, $_:b$ is a blank node):

```

⟨d:Poe, o:wrote, d:TheGoldBug⟩ ⟨d:Baudelaire, o:translated, d:TheGoldBug⟩
⟨d:Poe, o:wrote, d:TheRaven⟩ ⟨d:Mallarmé, o:translated, d:TheRaven⟩
⟨d:TheRaven, rdf:type, o:Poem⟩ ⟨d:Mallarmé, o:wrote, _:b⟩
⟨_:b, rdf:type, o:Poem⟩ ⟨d:TheGoldBug, rdf:type, o:Novel⟩

```

1. Draw an RDF graph corresponding to these statements
2. Express in English the meaning of these statements.

Chapter 3

Ontology languages

3.1 RDF Schema

What we considered so far is only the Simple RDF semantics. Full RDF is defined by identifying a particular vocabulary, the RDF vocabulary, and adding constraints to the definition of model in relation with this vocabulary. Because these changes are minor, we do not consider them and instead directly define RDFS Schema which contains full RDF.

RDFS (RDF Schema) [Brickley and Guha, 2004] is an extension of RDF designed to describe relationships between resources using a set of reserved words called the RDFS vocabulary. In the above example, the reserved word `rdf:type` can be used to relate instances to classes, e.g., `book1` is of type `publication`.

This section focusses on RDF and RDFS as extensions of the Simple RDF language presented in Section 2. Both extensions are defined in the same way:

- They consider a particular set of urirefs of the vocabulary prefixed by `rdf:` and `rdfs:`, respectively.
- They add additional constraints to the resources associated to these terms in interpretations.

In adding new constraints to RDFS interpretations, RDFS documents may have less models, and thus more consequences. It is possible for example, in RDF, to deduce $\langle \text{ex:author } \text{rdf:type } \text{rdf:Property} \rangle$ from $\langle \text{ex:person1 } \text{ex:author } \text{"Alkhateeb"} \rangle$; in RDFS, to deduce $\langle \text{ex:document1 } \text{rdf:type } \text{ex:Biography} \rangle$ from $\{ \langle \text{ex:document1 } \text{rdf:type } \text{rdf:Autobiography} \rangle, \langle \text{ex:Autobiography } \text{rdfs:subClassOf } \text{ex:Biography} \rangle \}$.

As usual, we first present the vocabulary (§3.1.1) and the constraints they put on the interpretation of the language (§3.1.2). We also introduce a “normative” inference mechanism (§3.1.3).

3.1.1 RDFS as an RDF vocabulary

In RDF and RDF Schema, there exists a set of reserved words, the RDF and RDFS vocabularies designed to describe relationships between resources like classes, e.g., `geo:City` `subClassOf` `geo:PopulatedArea`, and relationships between properties, e.g., `country` `subPropertyOf` `is-contained-in`. The RDFS vocabulary is presented in Table 3.1 as it appears in [Hayes, 2004]. The shortcuts that we will use for each of them are given in brackets.

From now on, we use *RDFSV* to denote the RDFS vocabulary.

	RDF Typing vocabulary	
rdf:type[type]	rdf:Property[prop]	rdf:XMLLiteral[xmlLit]
	Reification vocabulary	
rdf:Statement[stat]		
rdf:subject[subj]	rdf:predicate[pred]	rdf:object[obj]
	Container vocabulary	
rdf:first[first]	rdf:rest[rest]	rdf:nil[nil]
rdf: 1[1]	rdf: 2[2]	rdf: i[i]
rdf:List[list]	rdf:Bag[bag]	rdf:Seq[seq]
rdf:Alt[alt]		
	Miscellaneous vocabulary (do not use)	
rdf:value[value]		
	RDFS Typing vocabulary	
rdfs:Class[class]	rdfs:Resource[res]	rdfs:Literal[literal]
rdfs:domain[dom]	rdfs:range[range]	rdfs:Container[cont]
rdfs:subClassOf[sc]	rdfs:subPropertyOf[sp]	rdfs:Datatype[datatype]
	Container vocabulary	
rdfs:ContainerMembershipProperty[contMP]		rdfs:member[member]
	Documentation vocabulary	
rdfs:comment[comment]	rdfs:label[label]	rdfs:seeAlso[seeAlso]
	rdfs:isDefinedBy[isDefined]	

Table 3.1: The RDFS Vocabulary.

Example 3 (RDFS-graph). *Figure 3.1 displays the RDFS graph with the following assertions:*

```

geo:Grenoble department geo:Isere
departement rdfs:subPropertyOf is-contained-in
is-contained-in rdfs:range geo:GeographicArea
is-contained-in rdfs:domain geo:GeographicArea
    
```

3.1.2 RDFS semantics

The semantics of RDFS graphs is obtained by restricting the set of models of such graphs according to the specific vocabulary of the language.

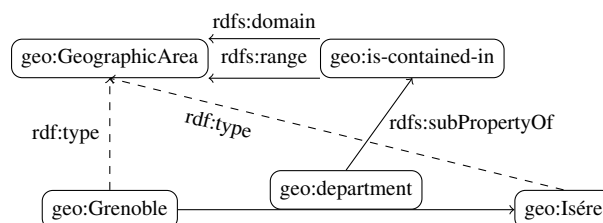


Figure 3.1: An RDFS graph. The arrows in dashed lines are consequences of the RDFS semantics.

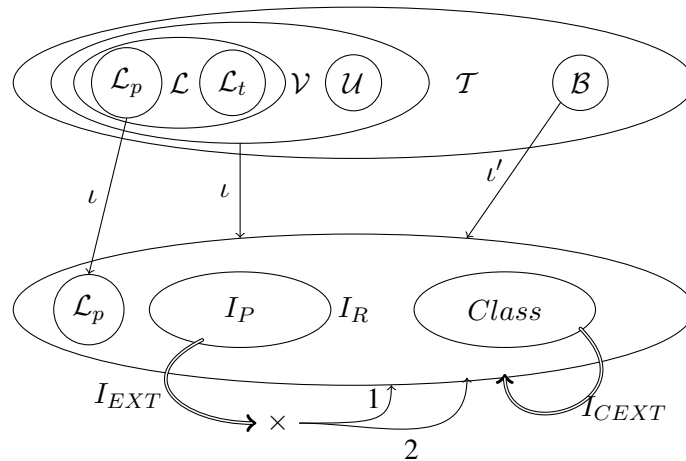


Figure 3.2: Domain structure for RDFS semantics.

In addition to the usual interpretation mapping, a special mapping is used in RDFS interpretations to allow interpreting the set of classes which is a subset of I_R .

Definition 10 (RDFS interpretation). *An RDFS interpretation of a vocabulary V is a tuple $\langle I_R, I_P, Class, I_{EXT}, I_{CEXT}, Lit, \iota \rangle$ such that:*

- $\langle I_R, I_P, I_{EXT}, \iota \rangle$ is an RDF interpretation;
- $Class \subseteq I_R$ is a distinguished subset of I_R identifying if a resource denotes a class of resources;
- $I_{CEXT} : Class \rightarrow 2^{I_R}$ is a mapping that assigns a set of resources to every resource denoting a class;
- $Lit \subseteq I_R$ is the set of literal values, Lit contains all plain literals in $\mathcal{L} \cap V$.

Specific conditions are added to the resources associated to terms of RDFS vocabularies in an RDFS interpretation to be an RDFS model of an RDFS graph. These conditions include the satisfaction of the RDF and RDF Schema axiomatic triples as appearing in the normative semantics of RDF [Hayes, 2004].

Definition 11 (RDF axiomatic triples). *RDF axiomatic triples are the triples in the following infinite set:*

$$\begin{aligned}
 &\langle type, type, prop \rangle \langle subj, type, prop \rangle \\
 &\quad \langle obj, type, prop \rangle \langle pred, type, prop \rangle \\
 &\langle first, type, prop \rangle \langle rest, type, prop \rangle \\
 &\langle value, type, prop \rangle \langle nil, type, list \rangle \\
 &\langle rdf:1, type, prop \rangle \langle rdf:2, type, prop \rangle \dots
 \end{aligned}$$

Definition 12 (RDFS axiomatic triples). *RDFS axiomatic triples are the triples in the following*

infinite set:

$$\begin{aligned}
 & \langle \text{type}, \text{dom}, \text{res} \rangle \langle \text{type}, \text{range}, \text{class} \rangle \\
 & \langle \text{dom}, \text{dom}, \text{prop} \rangle \langle \text{dom}, \text{range}, \text{class} \rangle \\
 & \langle \text{range}, \text{dom}, \text{prop} \rangle \langle \text{range}, \text{range}, \text{class} \rangle \\
 & \langle \text{sp}, \text{dom}, \text{prop} \rangle \langle \text{sp}, \text{range}, \text{prop} \rangle \\
 & \langle \text{sc}, \text{dom}, \text{class} \rangle \langle \text{sc}, \text{range}, \text{class} \rangle \\
 & \langle \text{subj}, \text{dom}, \text{stat} \rangle \langle \text{subj}, \text{range}, \text{res} \rangle \\
 & \langle \text{pred}, \text{dom}, \text{stat} \rangle \langle \text{pred}, \text{range}, \text{res} \rangle \\
 & \langle \text{obj}, \text{dom}, \text{stat} \rangle \langle \text{obj}, \text{range}, \text{res} \rangle \\
 & \langle \text{member}, \text{dom}, \text{res} \rangle \langle \text{member}, \text{range}, \text{res} \rangle \\
 & \langle \text{first}, \text{dom}, \text{list} \rangle \langle \text{first}, \text{range}, \text{res} \rangle \\
 & \langle \text{rest}, \text{dom}, \text{list} \rangle \langle \text{rest}, \text{range}, \text{list} \rangle \\
 & \langle \text{seeAlso}, \text{dom}, \text{res} \rangle \langle \text{seeAlso}, \text{range}, \text{res} \rangle \\
 & \langle \text{comment}, \text{dom}, \text{res} \rangle \langle \text{comment}, \text{range}, \text{literal} \rangle \\
 & \langle \text{isDefined}, \text{dom}, \text{res} \rangle \langle \text{isDefined}, \text{range}, \text{res} \rangle \\
 & \langle \text{label}, \text{dom}, \text{res} \rangle \langle \text{label}, \text{range}, \text{literal} \rangle \\
 & \langle \text{value}, \text{dom}, \text{res} \rangle \langle \text{value}, \text{range}, \text{res} \rangle \\
 \\
 & \langle \text{alt}, \text{sc}, \text{cont} \rangle \langle \text{bag}, \text{sc}, \text{cont} \rangle \qquad \langle \text{seq}, \text{sc}, \text{cont} \rangle \\
 & \langle \text{contMP}, \text{sc}, \text{prop} \rangle \\
 & \langle \text{isDefined}, \text{sp}, \text{seeAlso} \rangle \\
 & \langle \text{xmlLit}, \text{rdf:type}, \text{datatype} \rangle \langle \text{xmlLit}, \text{sc}, \text{literal} \rangle \\
 & \langle \text{datatype}, \text{sc}, \text{class} \rangle \\
 & \langle \text{rdf:-1}, \text{dom}, \text{res} \rangle \langle \text{rdf:-1}, \text{range}, \text{res} \rangle \qquad \langle \text{rdf:-1}, \text{rdf:type}, \text{contMP} \rangle \\
 & \langle \text{rdf:-2}, \text{dom}, \text{res} \rangle \langle \text{rdf:-2}, \text{range}, \text{res} \rangle \qquad \langle \text{rdf:-2}, \text{rdf:type}, \text{contMP} \rangle \\
 & \dots
 \end{aligned}$$

From this definition, it is clear that any RDFS interpretation is an RDF interpretation and that any RDF interpretation can be extended as different RDFS interpretations: it is sufficient to identify classes.

Definition 13 (RDFS Model). *Let G be an RDFS graph, and $I = \langle I_R, I_P, \text{Class}, I_{EXT}, I_{CEXT}, \text{Lit}, \iota \rangle$ be an RDFS interpretation of a vocabulary $V \subseteq \text{RDFSV} \cup \mathcal{V}$ such that $\mathcal{V}(G) \subseteq V$. Then I is an RDFS model of G if and only if I satisfies the following conditions:*

1. *Simple semantics:*

a) *there exists an extension ι' of ι to $\mathcal{B}(G)$ such that for each triple $\langle s, p, o \rangle$ of G , $\iota'(p) \in I_P$ and $\langle \iota'(s), \iota'(o) \rangle \in I_{EXT}(\iota'(p))$.*

2. *RDF semantics:*

a) $x \in I_P \Leftrightarrow \langle x, \iota'(\text{prop}) \rangle \in I_{EXT}(\iota'(\text{type}))$.

b) *If $\ell \in \text{term}(G)$ is a typed XML literal with lexical form w , then $\iota'(\ell)$ is the XML literal value of w , $\iota'(\ell) \in \text{Lit}$, and $\langle \iota'(\ell), \iota'(\text{xmlLit}) \rangle \in I_{EXT}(\iota'(\text{type}))$.*

c) *I satisfies all RDF axiomatic triples (Definition ??)*

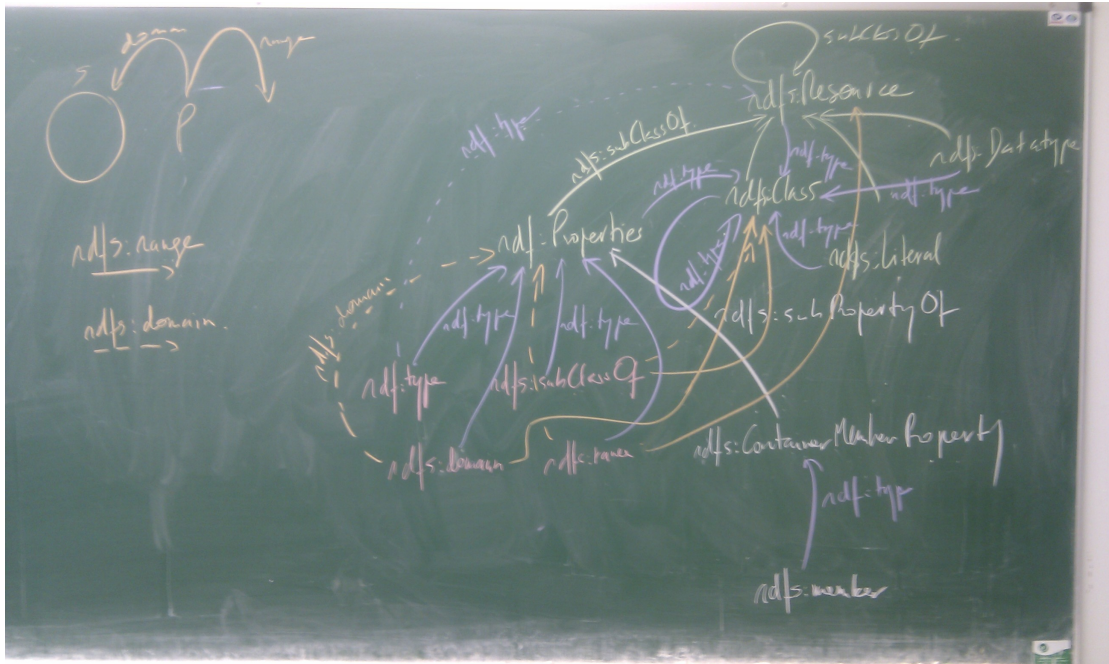


Figure 3.3: Part of the axiomatic triples performed live (exercise: find how many errors are there).

3. RDFS Classes:

- a) $x \in I_R, x \in I_{CEXT}(l'(res))$.
- b) $x \in Class, x \in I_{CEXT}(l'(class))$.
- c) $x \in Lit, x \in I_{CEXT}(l'(literal))$.

4. RDFS Subproperty:

- a) $I_{EXT}(l'(sp))$ is transitive and reflexive over I_P .
- b) if $\langle x, y \rangle \in I_{EXT}(l'(sp))$ then $x, y \in I_P$ and $I_{EXT}(x) \subseteq I_{EXT}(y)$.

5. RDFS Subclass:

- a) $I_{EXT}(l'(sc))$ is transitive and reflexive over $Class$.
- b) $\langle x, y \rangle \in I_{EXT}(l'(sc))$, then $x, y \in Class$ and $I_{CEXT}(x) \subseteq I_{CEXT}(y)$.

6. RDFS Typing:

- a) $x \in I_{CEXT}(y), \langle x, y \rangle \in I_{EXT}(l'(type))$.
- b) if $\langle x, y \rangle \in I_{EXT}(l'(dom))$ and $\langle u, v \rangle \in I_{EXT}(x)$ then $u \in I_{CEXT}(y)$.
- c) if $\langle x, y \rangle \in I_{EXT}(l'(range))$ and $\langle u, v \rangle \in I_{EXT}(x)$ then $v \in I_{CEXT}(y)$.

7. RDFS Additional:

- a) if $x \in Class$ then $\langle x, l'(res) \rangle \in I_{EXT}(l'(sc))$.
- b) if $x \in I_{CEXT}(l'(datatype))$ then $\langle x, l'(literal) \rangle \in I_{EXT}(l'(sc))$.
- c) if $x \in I_{CEXT}(l'(contMP))$ then $\langle x, l'(member) \rangle \in I_{EXT}(l'(sp))$.

d) *I satisfies all RDFS axiomatic triples (Definition ??)*

Any RDFS model is an RDF model. It is not true that any RDF model can support an RDFS model because the use of the RDFS vocabulary imposes additional constraints on RDFS models.

Definition 14 (RDFS consequence). *Let G and H be two RDFS graphs, then G RDFS-entails H (denoted by $G \models_{RDFS} H$) if and only if every RDFS model of G is also an RDFS model of H .*

[Hayes, 2004] states that “Since every RDFS interpretation is an RDF interpretation, if G RDFS-entails H then it RDF-entails H ” In the errata, the converse is given: “Since every RDFS interpretation is an RDF interpretation, if G RDF-entails H then it RDFS-entails H ” In fact, this does not seem immediate. It may be that G RDF-entails H , i.e., $\mathcal{M}_{RDF}(H) \subseteq \mathcal{M}_{RDF}(G)$ but we can only deduce that $\mathcal{M}_{RDFS}(H) \subseteq \mathcal{M}_{RDF}(H) \subseteq \mathcal{M}_{RDF}(G) \supseteq \mathcal{M}_{RDFS}(G)$, i.e., no relation between $\mathcal{M}_{RDFS}(H)$ and $\mathcal{M}_{RDFS}(G)$

Even the empty graph RDFS-entails far more assertions than it RDF-entails [Hayes, 2004], for example all triples of the form:

```
x rdf:type rdfs:Resource .
```

are true in all RDFS-interpretations of any vocabulary containing the URI reference x .

Example 4 (RDFS semantics). *Show why the graph of Example 3 RDFS-entails:*

```
geo:Isere rdf:type geo:GeographicArea
```

The same graph does not Simple RDF entails this assertion because there are RDF-models, which are not RDFS-models and which do not satisfy this triple (exhibit one). Does this holds for geo:Grenoble as well?

Example 5 (RDFS semantics). *Consider the graph G made of:*

```
ex:Pierre rdf:type ex:TennisPlayer
ex:TennisPlayer rdfs:subClassOf foaf:Person
ex:playsWith rdfs:domain ex:TennisPlayer
ex:defeated rdfs:subPropertyOf ex:playsWith
_:b ex:defeated ex:Pierre
```

$G \not\models_{RDF} \text{:}b \text{ rdf:type foaf:Person}$ because there is no subgraph of G that is an instance of this triple.

But $G \models_{RDFS} \text{:}b \text{ rdf:type foaf:Person}$ because for any model of G , $\langle \iota'(\text{:}b), \iota'(ex : Pierre) \rangle \in I_{EXT}(\iota'(ex : defeated))$ By 6c, this implies that $\iota'(\text{:}b) \in I_{CEXT}(\iota'(ex : TennisPlayer))$ In addition, $\langle \iota'(ex : TennisPlayer), \iota'(foaf : Person) \rangle \in I_{EXT}(\iota'(rdfs : subClassOf))$ By 5b, we have that $I_{CEXT}(\iota'(ex : TennisPlayer)) \subseteq I_{CEXT}(\iota'(foaf : Person))$ Hence, $\iota'(\text{:}b) \in I_{CEXT}(\iota'(foaf : Person))$

3.1.3 ter Horst closure

One possible approach for querying an RDFS graph G in a sound and complete way is by computing the closure graph of G , i.e., the graph obtained by saturating G with all information that can be deduced using a set of predefined rules called RDFS rules, then evaluating the query over the closure graph.

Definition 15 (RDFS closure). *Let G be an RDFS graph on an RDFS vocabulary V . The RDFS closure of G , denoted \hat{G} , is the smallest set of triples containing G and satisfying the following*

- [RDF1] all RDF axiomatic triples [Hayes, 2004] are in \hat{G} ;*
- [RDF2] if $\langle s, p, o \rangle \in \hat{G}$, then $\langle p, \text{type}, \text{prop} \rangle \in \hat{G}$;*
- [RDF3] if $\langle s, p, \ell \rangle \in \hat{G}$, where ℓ is an `xmlLit` typed literal and the lexical representation s is a well-formed XML literal, then $\langle s, p, \text{xml}(s) \rangle \in \hat{G}$ and $\langle \text{xml}(s), \text{type}, \text{xmlLit} \rangle \in \hat{G}$;*
- [RDFS 1] all RDFS axiomatic triples [Hayes, 2004] are in \hat{G} ;*
- [RDFS 6] if $\langle a, \text{dom}, x \rangle \in \hat{G}$ and $\langle u, a, y \rangle \in \hat{G}$, then $\langle u, \text{type}, x \rangle \in \hat{G}$;*
- [RDFS 7] if $\langle a, \text{range}, x \rangle \in \hat{G}$ and $\langle u, a, v \rangle \in \hat{G}$, then $\langle v, \text{type}, x \rangle \in \hat{G}$;*
- [RDFS 8a] if $\langle x, \text{type}, \text{prop} \rangle \in \hat{G}$, then $\langle x, \text{sp}, x \rangle \in \hat{G}$;*
- [RDFS 8b] if $\langle x, \text{sp}, y \rangle \in \hat{G}$ and $\langle y, \text{sp}, z \rangle \in \hat{G}$, then $\langle x, \text{sp}, z \rangle \in \hat{G}$;*
- [RDFS 9] if $\langle a, \text{sp}, b \rangle \in \hat{G}$ and $\langle x, a, y \rangle \in \hat{G}$, then $\langle x, b, y \rangle \in \hat{G}$;*
- [RDFS 10] if $\langle x, \text{type}, \text{class} \rangle \in \hat{G}$, then $\langle x, \text{sc}, \text{res} \rangle \in \hat{G}$;*
- [RDFS 11] if $\langle u, \text{sc}, x \rangle \in \hat{G}$ and $\langle y, \text{type}, u \rangle \in \hat{G}$, then $\langle y, \text{type}, x \rangle \in \hat{G}$;*
- [RDFS 12a] if $\langle x, \text{type}, \text{class} \rangle \in \hat{G}$, then $\langle x, \text{sc}, x \rangle \in \hat{G}$;*
- [RDFS 12b] if $\langle x, \text{sc}, y \rangle \in \hat{G}$ and $\langle y, \text{sc}, z \rangle \in \hat{G}$, then $\langle x, \text{sc}, z \rangle \in \hat{G}$;*
- [RDFS 13] if $\langle x, \text{type}, \text{contMP} \rangle \in \hat{G}$, then $\langle x, \text{sp}, \text{member} \rangle \in \hat{G}$;*
- [RDFS 14] if $\langle x, \text{type}, \text{datatype} \rangle \in \hat{G}$, then $\langle x, \text{sc}, \text{literal} \rangle \in \hat{G}$.*

constraints:

It is easy to show that this closure always exists and can be obtained by turning the constraints into rules, thus defining a closure operation.

Example 6 (RDFS Closure). *The RDFS closure of the RDFS graph of Example 3 contains, in particular, the following assertions:*

```
is-contained-in rdf:type rdf:Property // [RDF 2]
departement rdf:type rdf:Property // [RDF 2]
geo:Grenoble is-contained-by geo:Isere // [RDFS 9]
geo:Grenoble rdf:type geo:GeographicArea // [RDFS 6]
geo:Isere rdf:type geo:GeographicArea // [RDFS 7]
...
```

Because of axiomatic triples, this closure may be infinite, but a finite and polynomial closure, called *partial closure*, has been proposed independently in [Baget, 2003] and [ter Horst, 2005].

Definition 16 (Partial RDFS closure). *Let G and H be two RDFS graphs on an RDFS vocabulary V , the partial RDFS closure of G given H , denoted $\hat{G}\setminus H$, is obtained in the following way:*

1. *let k be the maximum of i 's such that $\text{rdf}:_i$ is a term of G or of H ;*
2. *replace the rule [RDF 1] by the rule [RDF 1P] add all RDF axiomatic triples [Hayes, 2004] except those that use $\text{rdf}:_i$ with $i > k$. In the same way, replace the rule [RDFS 1] by the rule [RDFS 1P] add all RDFS axiomatic triples except those that use $\text{rdf}:_i$ with $i > k$;*
3. *apply the modified rules.*

Applying closure to an RDFS graph permits to reduce RDFS entailment to simple RDF entailment.

Proposition 5 (Completeness of partial RDFS closure [Hayes, 2004]). *Let G be a satisfiable RDFS graph and H an RDFS graph, then $G \models_{RDFS} H$ if and only if $(\hat{G}\setminus H) \models_{RDF} H$.*

$\hat{G}\setminus H$ is limited to some consequences: those that may affect H , hence this completeness is relative to H . The completeness does not hold if G is not satisfiable because in such a case, any graph H is a consequence of G and \models_{RDF} does not reflect this (no RDF graph can be inconsistent). An RDFS graph can be unsatisfiable only if it contains datatype conflicts [ter Horst, 2005] which can be found in polynomial time.

3.1.4 Computational properties

We can define the RDFS ENTAILMENT problem in the same way as we defined SIMPLE RDF ENTAILMENT:

RDFS ENTAILMENT

Instance: two RDFS graphs G and H .

Question: Does $G \models_{RDFS} H$?

RDFS ENTAILMENT is an NP-complete problem [Gutierrez *et al.*, 2004].

3.1.5 Conclusion

Full RDF extends simple RDF with a specific vocabulary. Similarly, RDF Schema extend RDF with another vocabulary. These vocabularies express constraints on the entities in RDF graphs: their type and structure for RDF and further constraints on domains of properties and class/property specialisation. These vocabularies bring specific constraints to the semantics. These constraints further extend the definition of RDFS interpretations, and reduces the interpretations which are models of RDF and RDFS graphs. Hence the notion of consequence is different with Full RDF or RDFS graph.

It is not possible anymore to reduce entailment to the existence of a homomorphism. However, by computing a closure of the entailing graph, it is possible to generate a larger graph against which RDFS entailment correspond to simple RDF entailment (and thus homomorphism

testing is again complete and correct). Although, the closure may be infinite, it is possible to compute a smaller, finite useful subset called the partial closure against which homomorphism checking is again complete and correct.

3.2 The web ontology language OWL

RDF and RDFS allows to assert relations between classes and properties, e.g., `subClassOf`, but they do not allow to construct these from the inside. OWL [Dean and Schreiber, 2004] is the language for constructing classes and properties. It is strongly inspired from description logics [Baader *et al.*, 2003]. The W3C has released a new version of OWL (OWL 2.0) [Beckett, 2009a].

We present below the syntax, semantics and the different standard sublanguages of OWL.

3.2.1 OWL syntax

OWL is more than a vocabulary: there are many RDF graphs with OWL vocabulary which cannot be interpreted in OWL. It is more reasonable to think that OWL has a syntax, cast in RDF syntax, in the more classic sense of the term.

In particular, OWL abstractly defines *class descriptions* and *property descriptions*. These can be defined recursively like terms

Definition 17 (OWL terminology). *In OWL, the terminology is still divided among literals \mathcal{L} , URI references \mathcal{U} and variables \mathcal{B} . However, the set \mathcal{U} is divided into many sets $(\langle \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle)$:*

\mathcal{C} the set of class names containing `owl:Thing` and `owl:Nothing`;

\mathcal{D} the set of data types containing `rdfs:Literal`;

\mathcal{I} the set of individual names;

\mathcal{P}_d the set of data type property names;

\mathcal{P}_i the set of object property names;

In the initial definition, there were also others categories:

$\mathcal{P}_a = V_{AP}$ the set of annotation property names containing `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`, `rdfs:versionInfo` and `rdfs:isDefinedBy`. These have been renamed *facets* in OWL 2;

$\mathcal{O} = V_O$ the set of ontology names. They have been dropped in OWL 2 semantics.

but, in the following, we will only consider $\mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i$ since these are the only ones with a meaningful semantics. We will also ignore documentation vocabulary.

The OWL vocabulary is given in Table 3.2 following [Dean and Schreiber, 2004] for OWL and [Bao *et al.*, 2009] for OWL 2. We only cover the main terms, miscellaneous one are not considered here. One of the main difference between OWL and OWL 2 is a uniform treatment of datatypes which is also lightly covered here.

Example 7. *The following example:*

$$\begin{aligned} \text{ChemistryProfessor} &\sqsubseteq \text{Professor} \sqcap \forall \text{teaches} . \text{ChemistryLecture} \\ \text{teaches} &= \text{taughtBy}^{-1} \\ \top &\sqsubseteq \forall \text{teaches} . \text{Lecture} \sqcap \forall \text{teaches}^{-1} . \text{Professor} \end{aligned}$$

	Typing vocabulary	
owl:Class	owl:Thing	owl:Nothing
rdf:Property	owl:Data(type)Property	owl:ObjectProperty
owl:topObjectProperty ²	owl:bottomObjectProperty ²	owl:topDataProperty ²
owl:bottomDataProperty ²		
	Class/Datatype⁺ constructor	
owl:intersectionOf ⁺	owl:unionOf ⁺	owl:complementOf
owl:oneOf ⁺	owl:datatypeComplementOf ²	owl:disjointUnionOf ^{2?}
	Property restrictions	
owl:Restriction	owl:onProperty	owl:onClass ²
owl:someValuesFrom ⁺	owl:minQualifiedCardinality ²	owl:minCardinality
owl:allValuesFrom ⁺	owl:maxQualifiedCardinality ²	owl:maxCardinality
owl:hasValue	owl:qualifiedCardinality ²	owl:cardinality
	Class constraints	
owl:equivalentClass	rdfs:subClassOf	owl:disjointWith
owl:hasSelf ²	owl:hasKey ²	
	Property constraints	
owl:equivalentProperty	rdfs:subPropertyOf	owl:inverseOf
	owl:propertyChainAxiom	
owl:FunctionalProperty	owl:TransitiveProperty	owl:SymmetricProperty
owl:AsymmetricProperty ²	owl:ReflexiveProperty ²	owl:IreflexiveProperty ²
rdfs:domain	owl:InverseFunctionalProperty	rdfs:range
	Individual constraints	
	owl:sameAs	owl:differentFrom
	Global constraints	
owl:AllDifferent	owl:AllDisjointproperties ²	owl:AllDisjointClasses ²
	owl:Ontology	owl:imports
	Documentation	
owl:versionInfo	owl:priorVersion	
owl:DeprecatedClass	owl:backwardCompatibleWith	
owl:DeprecatedProperty	owl:AnnotationProperty	owl:OntologyProperty

Table 3.2: The OWL vocabulary. OWL 2 primitives are followed by an "2" exponent; primitives applicable to datatypes since OWL 2 are marked with a "+" exponent.

can be expressed in OWL by:

```
<owl:Class rdf:about="#ChemistryProfessor">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Professor" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="#teaches" />
          <owl:allValuesFrom rdf:resource="#ChemistryLecture" />
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:about="#teaches">
  <owl:inverseOf rdf:resource="#taughtBy" />
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range rdf:resource="#Lecture" />
</owl:ObjectProperty>
```

*It is noteworthy that the last statements are an encoding of `rdfs:domain` and `rdfs:range`.
The same can be expressed in the terrible syntax of triples:*

```
ex:ChemistryProfessor rdf:type owl:Class
ex:ChemistryProfessor rdfs:subClassOf _:b1
_:b1 rdf:type owl:Class
_:b1 owl:intersectionOf _:b2
_:b2 rdf:type rdfs:Collection
_:b2 rdf:_1 ex:Professor
ex:Professor rdf:type owl:Class
_:b2 rdf:_2 _:b3
_:b3 rdf:type owl:Restriction
_:b3 owl:onProperty ex:teaches
_:b3 owl:allValuesFrom ex:ChemistryLecture
```

3.2.2 OWL semantics

The semantics of OWL constructs is given in [Patel-Schneider *et al.*, 2004; Motik *et al.*, 2009b] following a description logic style.

Definition 18 (OWL interpretation). *An OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$ is a tuple $I = \langle I_R, E_C, E_R, L, S, V, O \rangle$ such that:*

- I_R (resources) $\neq \emptyset$;
- $O \neq \emptyset$;
- $O \subseteq I_R$;
- $O \cap V = \emptyset$;
- $V \subseteq I_R$ (plus extras)
- $E_C : \mathcal{C} \rightarrow 2^O$;
- $E_C : \mathcal{D} \rightarrow 2^V$;

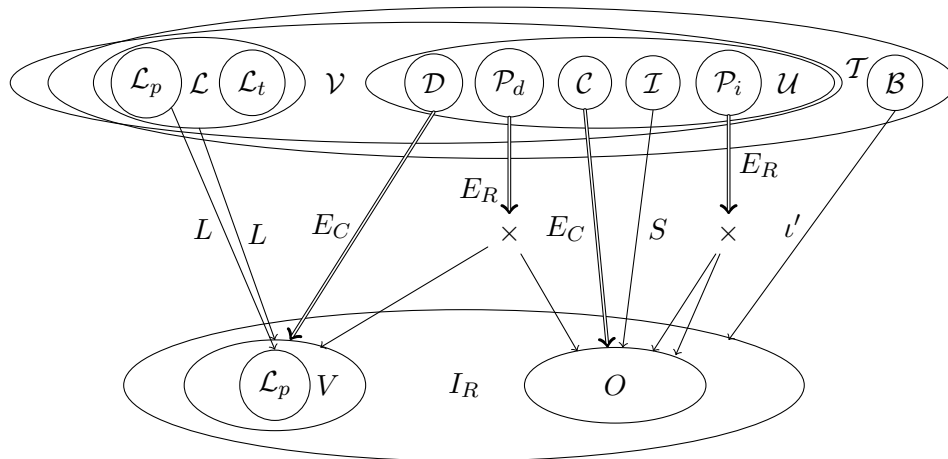


Figure 3.4: Domain structure for OWL semantics.

- $E_R : \mathcal{P}_d \rightarrow 2^{O \times V}$;
- $E_R : \mathcal{P}_i \rightarrow 2^{O \times O}$;
- $L : \mathcal{L} \rightarrow V$ (plus extra):
- $S : \mathcal{C} \cup \mathcal{D} \cup \mathcal{I} \cup \mathcal{P}_d \cup \mathcal{P}_i \rightarrow I_R$ (plus extra)
- $S(\mathcal{I}) \subseteq O$;
- more about datatypes

Compound expressions are interpreted according to the classical description logic semantics.

Definition 19 (Interpretation of compound expressions). *Let $I = \langle I_R, E_C, E_R, L, S, V, O \rangle$ be an OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$ the interpretation of compound*

class expressions by I is defined by (all keywords are in the OWL namespace):

$$\begin{aligned}
 E_C(\text{owl:Thing}) &= O \\
 E_C(\text{owl:Nothing}) &= \emptyset \\
 E_C(\text{rdfs:Literal}) &= V \\
 E_C(\text{complementOf}(c)) &= O - E_C(c) \\
 E_C(\text{unionOf}(c, c')) &= E_C(c) \cup E_C(c') \\
 E_C(\text{intersectionOf}(c, c')) &= E_C(c) \cap E_C(c') \\
 E_C(\text{oneOf}(i_1, \dots, i_n)) &= \{S(i_1), \dots, S(i_n)\} \\
 E_C(\text{oneOf}(v_1, \dots, v_n)) &= \{L(i_1), \dots, L(i_n)\} \\
 E_C(\text{restriction}(p, \text{allValuesFrom}(r))) &= \{x \in O; \forall \langle x, y \rangle \in E_R(p), y \in E_C(r)\} \\
 E_C(\text{restriction}(p, \text{someValueFrom}(r))) &= \{x \in O; \exists \langle x, y \rangle \in E_R(p) \wedge y \in E_C(r)\} \\
 E_C(\text{restriction}(p, \text{hasValue}(v))) &= \{x \in O; \langle x, L(v) \rangle \in E_R(p)\} \\
 E_C(\text{restriction}(p, \text{hasValue}(i))) &= \{x \in O; \langle x, S(i) \rangle \in E_R(p)\} \\
 E_C(\text{restriction}(p, \text{minCardinality}(n))) &= \{x \in O; |\{\langle x, y \rangle \in E_R(p)\}| \geq n\} \\
 E_C(\text{restriction}(p, \text{maxCardinality}(n))) &= \{x \in O; |\{\langle x, y \rangle \in E_R(p)\}| \leq n\} \\
 E_C(\text{restriction}(p, \text{cardinality}(n))) &= \{x \in O; |\{\langle x, y \rangle \in E_R(p)\}| = n\} \\
 E_R(\text{inverseOf}(p)) &= \{\langle x, y \rangle \in O \times O; \langle y, x \rangle \in E_R(p)\}
 \end{aligned}$$

Like for description logics, it would be possible to interpret other type of compound expressions (like property path expressions).

Definition 20 (Axiom satisfaction). *Let $I = \langle I_R, E_C, E_R, L, S, V, O \rangle$ be an OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$ an axiom is said to be satisfied (all keywords are in the OWL namespace):*

$$\begin{aligned}
 \text{equivalentClass}(c, c') &\text{ iff } E_C(c) = E_C(c') \\
 \text{subClassOf}(c, c') &\text{ iff } E_C(c) \subseteq E_C(c') \\
 \text{disjointWith}(c, c') &\text{ iff } E_C(c) \cap E_C(c') = \emptyset \\
 \text{equivalentProperty}(p, p') &\text{ iff } E_R(p) = E_R(p') \\
 \text{subPropertyOf}(p, p') &\text{ iff } E_R(p) \subseteq E_R(p') \\
 \text{sameAs}(i, i') &\text{ iff } S(i) = S(i') \\
 \text{differentFrom}(i, i') &\text{ iff } S(i) \neq S(i') \\
 \text{AllDifferent}(i_0, \dots, i_n) &\text{ iff } \forall j, k \in [0, n], j \neq k \Rightarrow S(i_j) \neq S(i_k) \\
 \text{SymmetricProperty}(p) &\text{ iff } \forall \langle x, y \rangle \in E_R(p), \langle y, x \rangle \in E_R(p) \\
 \text{TransitiveProperty}(p) &\text{ iff } \forall \langle x, y \rangle, \langle y, z \rangle \in E_R(p), \langle x, z \rangle \in E_R(p) \\
 \text{FunctionalProperty}(p) &\text{ iff } \forall \langle x, y \rangle, \langle x, z \rangle \in E_R(p), y = z \\
 \text{InverseFunctionalProperty}(p) &\text{ iff } \forall \langle y, x \rangle, \langle z, x \rangle \in E_R(p), y = z
 \end{aligned}$$

A model is defined as usual as an interpretation that satisfies all axioms. It also satisfies the constraints raised by the category of each term.

Definition 21 (OWL model). *An OWL interpretation $I = \langle I_R, E_C, E_R, L, S, V, O \rangle$ over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$ is a model of an OWL Ontology \mathcal{O} iff*

- each *uriref* used in class (resp. datatype property, object property, individual, datatype) position in \mathcal{O} , belongs to \mathcal{C} (resp. $\mathcal{P}_d, \mathcal{P}_i, \mathcal{I}, \mathcal{D}$);
- each literal in \mathcal{O} belongs to \mathcal{L} ;
- I satisfies all axioms in \mathcal{O} .

Consequence, inconsistency, etc. are defined as usual.

Example 8. Here is an example of an OWL ontology in the terrible syntax of a set of triples:

```
ex:Grenoble rdf:type geo:City
geo:City rdfs:subClassOf geo:GeographicArea
ex:is-contained-in owl:inverseOf ex:contains
geo:City rdfs:subClassOf _:x
_:x rdf:type owl:Restriction
_:x owl:onProperty ex:is-contained-in
_:x owl:maxCardinality "1"^^xsd:Integer
geo:City rdfs:subClassOf _:y
_:y rdf:type owl:Restriction
_:y owl:onProperty ex:country
_:y owl:someValuesFrom geo:Country
```

It entails the following sets of triples:

```
ex:Grenoble rdf:type geo:GeographicArea
ex:Isere ex:contains ex:Grenoble
ex:Grenoble ex:country _:z
_:z rdf:type ex:Country
```

Moreover, if one adds:

```
ex:Grenoble ex:country ex:France
ex:country rdfs:subPropertyOf ex:is-contained-in
```

this entails:

```
ex:France owl:sameAs ex:Isere
```

which is obviously wrong. So what is the problem?

Example 9. Consider the ontology made of the following statements:

$$\begin{aligned} \text{ChemistryProfessor} &\sqsubseteq \text{ScienceProfessor} \sqcap \forall \text{teaches.ChemistryLecture} \\ \text{ScienceProfessor} &\sqsubseteq \text{Professor} \sqcap \exists \text{teaches.ScienceLecture} \end{aligned}$$

The OWL semantics makes that: $\models_{OWL} \text{ChemistryProfessor} \sqsubseteq \text{Professor}$ holds because for any model,

$$\begin{aligned} E_C(\text{ChemistryProfessor}) &\subseteq E_C(\text{ScienceProfessor} \sqcap \forall \text{teaches.ChemistryLecture}) \\ &= E_C(\text{ScienceProfessor}) \cap E_C(\forall \text{teaches.ChemistryLecture}) \\ &= E_C(\text{ScienceProfessor}) \cap \{o \mid \forall y; \langle o, y \rangle \in E_R(\text{teaches}), y \in E_C(\text{ChemistryLecture})\} \\ E_C(\text{ScienceProfessor}) &\subseteq E_C(\text{Professor} \sqcap \exists \text{teaches.ScienceLecture}) \\ &= E_C(\text{Professor}) \cap E_C(\exists \text{teaches.ScienceLecture}) \\ &= E_C(\text{Professor}) \cap \{o \mid \exists y; \langle o, y \rangle \in E_R(\text{teaches}), y \in E_C(\text{ChemistryLecture})\} \end{aligned}$$

Hence, $E_C(\text{ChemistryProfessor}) \subseteq E_C(\text{ScienceProfessor}) \subseteq E_C(\text{Professor})$ is true in all models.

It is not true that $\models_{OWL} \text{.}b \text{ rdf:type ex:ChemistryLecture}$. However, if one adds the statement that $\text{ex:Paul rdf:type ex:ChemistryProfessor}$, then $\models_{OWL} \text{.}b \text{ rdf:type ex:ChemistryLecture}$.

OWL in general also enjoys infinite closure (for instance, a maximum cardinality restriction entails maximum cardinality restriction of all superior integers), but the patterns of entailed statements are wider. Hence, it is not advised to use closure.

Inference in OWL is usually provided by dedicated description logic provers based on the tableau method, resolution or other transformations. Most often they proceed by refutation. In order to prove that $G \models_{OWL} H$, instead of investigating all models of G for checking that H holds in all of them, they starts with G and an encoding of the negation of H and tries to build *one* model satisfying them. If such model is found this means that there exist a model of G not satisfying H , so H is not entailed by G . If no such model is found, then this means that H holds in all models of H .

3.2.3 OWL 2 and profiles

The initial OWL specification defined three different sublanguages (OWL Full, OWL DL and OWL Lite). Fortunately people realised that the framework of description logics was rich enough to accommodate many more languages. We consider here the state of the OWL specifications. In addition, of numerous minor changes, OWL 2 introduces the notion of profiles [Motik *et al.*, 2009a], reminiscent of description logic modularity which describe a particular subset of OWL. These profiles cover the previous OWL sublanguages and introduce three new OWL 2 profiles. A profile is characterised by the vocabulary it uses and the syntactic form of the graphs it accepts.

These are:

- OWL Lite** – has separate domains of discourse for properties and classes;
- contains all RDF constructors, i.e., it allows to declare an individual as member of a class and to relate individuals and data values through properties;
 - uses part of the RDFS vocabulary (`rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:range`, `rdfs:domain`), with the same semantics;
 - enables the definition of new classes (`owl:Class`) as more specific or equivalent to the conjunction of other classes;
 - `owl:sameIndividualAs` and `owl:differentIndividualFrom` assert that two individuals are equivalent or different;
 - properties characteristics can be asserted through: `owl:inverseOf` states that a property p is the converse of another p' (hence, the triple $\langle s p o \rangle$ entails $\langle o p' s \rangle$); other characteristics are transitivity (`owl:TransitiveProperty`), functionality (`owl:FunctionalProperty`), symmetry (`owl:SymmetricProperty`) or inverse functionality (`owl:InverseFunctionalProperty`),
 - `owl:allValuesFrom` defines the class of objects for which all values to property p belong to class c ; `owl:someValuesFrom` defines the class of objects for which there exists a value to property p belonging to class c ;
 - `owl:minCardinality` (resp. `owl:maxCardinality`) defines the class of objects which have at least (resp. at most) n values for property p . In OWL Lite, n can

only be 0 or 1.

OWL Lite corresponds to the $\mathcal{SHIF}(\mathcal{D})$ description logic (concept conjunction, disjunction and negation+universal role restriction+qualified existential role restriction+transitive roles+role hierarchy+inverse roles+functional roles+datatypes). $\mathcal{SHIF}(\mathcal{D})$ entailment can be decided in EXPTIME [Horrocks *et al.*, 2003] and there are efficient algorithms for solving the problem.

OWL 2 RL is a fragment of OWL Full that can be implemented with a rule language.

OWL 2 QL corresponds roughly to DL-Lite (§3.2.4);

OWL DL contains all constructors, with particular constraints on their use that warrants the decidability of entailment tests. So, OWL DL:

- Contains all OWL Lite constructors;
- Allows any positive integer in cardinality constraints but prohibits them on transitive properties or their inverse or any of their subproperties;
- `owl:oneOf` defines a class through the set of its instances,
- `owl:hasValue` defines the class of objects having only a particular value v for some property p ;
- `owl:disjointWith` asserts that two classes have no common instances;
- `owl:unionOf` and `owl:complementOf` allow the definition of a class as the union of two classes or the complement of one class.

OWL DL corresponds to the $\mathcal{SHOIN}(\mathcal{D})$ description logic ($\mathcal{SHIF}(\mathcal{D})$ +nominals). $\mathcal{SHOIN}(\mathcal{D})$ entailment test has a non deterministic exponential time complexity (NEXPTIME) [Horrocks *et al.*, 2003]. So far there is no known complete algorithm for this problem that can be implemented.

OWL 2 EL retains an expressive power in the definition of classes and properties restricted to existential quantification (it drops universal quantification, cardinality restrictions, disjunction, negation and relation properties);

OWL Full uses all constructors without any constraint. It does not correspond to a well identified class of description logics due to its non standard semantics. Since it contains negation and definition of sets which can contain defined sets, it is most likely undecidable.

- Contains all OWL DL constructors;
- Contains all RDF Schema constructors;
- Allows to use classes in place of individuals in constructors.

Inference engines have been implemented for significant subsets of OWL DL (in particular Pellet). Moreover there are still works for finding sweeter spots in the OWL language family. One of these is DL-Lite that is presented below.

3.2.4 DL-Lite a sub family on its own

DL-Lite [Calvanese *et al.*, 2007] is a family of description logics that has been designed for supporting efficient (polynomial) conjunctive query answering. It is based on the reduction of the expressivity of the OWL language. This is achieved by choosing sublanguages whose query evaluation can be reduced to simple relational database queries.

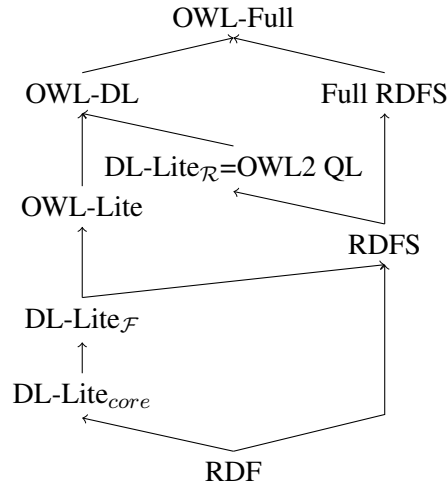


Figure 3.5: A hierarchy of OWL languages.

Definition 22 (DL-Lite syntax). *DL-Lite terms are based on the following grammar:*

$$\begin{aligned} C &\rightarrow B \mid \neg B \\ B &\rightarrow A \mid \exists R \\ R &\rightarrow P \mid P^{-1} \end{aligned}$$

in which A and P are URI identifying concepts and properties respectively. The $DL-Lite_{core}$ language contains assertions of the form $B \sqsubseteq C$, in addition, $DL-Lite_{\mathcal{R}}$ authorises axioms like $R \sqsubseteq R$ and $R \sqsubseteq \neg R$, and $DL-Lite_{\mathcal{F}}$ added functionality axioms.

This corresponds to the use of the following OWL constructs: `rdfs:subClassOf`, `owl:minCardinality` (with value 1), `owl:Thing`, `owl:Nothing`, `owl:complementOf`, `owl:inverseOf`, to which is added the complement of a property. This simple language allows to express assertions like $B \sqsubseteq C \sqcap C'$ which is equivalent to $B \sqsubseteq C$ and $B \sqsubseteq C'$.

In fact, $DL-Lite_{\mathcal{F}}$ is a strict subset of OWL-Lite or RDFS (it cannot express `rdfs:range` or `owl:allValuesFrom`) and apparently $DL-Lite_{\mathcal{R}}$ is a strict superset of (the description logic part of) RDFS.

DL-Lite languages comply with the semantics of OWL, but for one thing: they obey the unique name assumption, hence it is not possible that two URIs identify the same individual.

Example 10. *The example given in Marie-Christine Rousset's slides [Abiteboul et al., 2011]:*

$$\begin{aligned} &Professor \sqsubseteq \exists teachesTo \\ &Student \sqsubseteq \exists hasTutor \\ \exists teachesTo^{-1} &\sqsubseteq Student \\ \exists hasTutor^{-1} &\sqsubseteq Professor \\ Professor &\sqsubseteq \neg Student \\ hasTutor^{-1} &\sqsubseteq teachesTo && DL - Lite_{\mathcal{R}} \\ &(func\ hasTutor) && DL - Lite_{\mathcal{F}} \end{aligned}$$

can be expressed in OWL by:

```
<owl:Class rdf:about="#Professor">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teachesTo" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Student" />
</owl:Class>

<owl:Class rdf:about="#Student">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTutor" />
      <owl:someValuesFrom rdf:resource="#&owl;Thing" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty>
          <owl:inverseOf rdf:resource="#teachesTo" />
        </owl:ObjectProperty>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#&owl;Thing">
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Student" />
</owl:Class>

<owl:Class>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty>
        <owl:Property>
          <owl:inverseOf rdf:resource="#hasTutor" />
        </owl:Property>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#&owl;Thing">
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Professor" />
</owl:Class>

<owl:Property rdf:about="#hasTutor">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty>
      <owl:inverseOf rdf:resource="#teachesTo" />
    </owl:ObjectProperty>
  </rdfs:subPropertyOf>
</owl:Property>
```

Problem	Axiom entailment	Conjunctive query
OWL Full	undecidable	undecidable
OWL 2	2NEXPTIME-complete	open
OWL 1 DL	NEXPTIME-complete	open (decidable)
OWL 2 EL	PTIME-complete	PSPACE-complete
OWL 2 RL	PTIME-complete	NP-complete
OWL 2 QL	NLOGSPACE-complete	NP-complete (UCQ)
RDFS		
OWL 1 Lite		
DL-Lite _{\mathcal{F}}	PTIME	NP-complete (UCQ)
DL-Lite _{core}	PTIME	

Table 3.3: Worst-case combined complexity results for problems in OWL.

```

</owl:ObjectProperty>
</rdfs:subPropertyOf>
</owl:Property>

<owl:FunctionalProperty rdf:about="#hasTutor" />

```

The definition of *hasTutor* is correct because if $r \sqsubseteq r'$, then $r^{-1} \sqsubseteq r'^{-1}$ (prove it and see how it affect our definition).

You can as well try to express this as a set of triples and check that the semantics for this piece of description logic is the same as the semantics for OWL.

3.2.5 Computational results

Table 3.3 summarises the results from [Motik *et al.*, 2009a] and [Calvanese *et al.*, 2007] about the complexity of the axiom entailment problem (consistency, class satisfiability, class subsumption, instance checking) and conjunctive query answering.

Because we retained combined complexity, the choice of a particular logic may also depend on the dominant factor in the problem to be dealt with (the size of the data, the size of the query or that or the ontology). Of course, this choice should primarily depend on what is to be expressed.

3.2.6 Provisory conclusion

Although, RDF Schema allows for defining simple ontologies, it does not allow for powerful constraints such as defining classes by disjunction or complement or by constraining their properties on type or cardinality. The OWL language introduces further vocabulary (and structure constraints) for defining more expressive ontologies. This led to extend further the semantic structure of these languages.

3.3 Conclusion

3.4 Exercises

Exercise 4 (OWL ontologies).

1. Describe in OWL (RDF or XML/RDF) the ontology containing the following assertions:
 - All authors are persons;
 - A book ($m:livre$) has exactly one year of publication ($m:annee$);
 - A novel ($m:roman$) is a book ($m:livre$) and a book is a work ($m:oeuvre$);
 - The title ($dc:title$) of a work is a character string ($xsd:string$);
 - The relation "a écrit" ($m:aecrit$) relates an author to a work.
2. If one associates the graph (a) of Figure 2.5 of Exercise 1 (p20) and the ontology resulting from the previous question, is it possible to deduce the type ($rdf:type$) of $?x$?
Can you semantically justify how? What else can be deduced?
3. Does the use of this ontology for defining the graphs of Figure 2.5 (p20) would change something to the answer to Question 3 of Exercise 1? What?

Exercise 5 (DL-Lite ontologies). Consider the ontology O made of the following DL-Lite assertions:

$$\begin{aligned} worksWith &\sqsubseteq foaf:knows \\ playsTennisWith &\sqsubseteq playsSportWith \\ playsSportWith &\sqsubseteq foaf:knows \\ marriedWith &\sqsubseteq foaf:knows \\ Person &\sqsubseteq \exists foaf:mbox \end{aligned}$$

1. In which dialect (sublanguage) of DL-Lite is this ontology expressed ($DL-Lite_{core}$, $DL-Lite_{\mathcal{F}}$, $DL-Lite_{\mathcal{R}}$)?
2. Rewrite O in OWL or RDFS.
3. In which dialect (sublanguage) of RDFS or OWL is your ontology expressed?
4. Given the graph of Figure 2.6(b) of Exercise 2 (p21) to which the axioms of O are added. Compute its closure. What is the difference with the partial closure?
5. Given the graphs of Figure 2.6 (p21) to which the axioms of O are added, does this change something to the answers given to Question 3 of Exercise 2 if we consider RDFS-entailment? (explain why)

Exercise 6 (OWL ontologies).

1. Provide an ontology satisfied by both graphs of Figure 2.6 of Exercise 2 (p21).

We would like to express that an email address cannot correspond to more than one person by the property `foaf:mailbox`.

2. How to express this in OWL: with a cardinality constraint, a functional property, an inverse functional property or a symmetric property?
3. Is it possible to express this constraint in DL-Lite? If yes, how?
4. Consider the graphs of Figure 2.6 (p21) to which this constraint is added, does this change something to the answers given to Question 3 of Exercise 2? (explain why)

Exercise 7 (RDFS ontologies). Consider the RDFS ontology o containing, in addition to those of the graph G of Exercise 3, the following statements:

```
⟨o:Novel, rdfs:subClassOf, o:Literature⟩  
⟨o:Poem, rdfs:subClassOf, o:Literature⟩  
⟨o:translated, rdfs:range, o:Literature⟩  
⟨o:wrote, rdfs:domain, o:Writer⟩
```

1. Does this allow to conclude that `d:Poe`, `d:Baudelaire` or `d:Mallarmé` is a `o:Writer`? Explain why.
2. Can you express in OWL the statement that “anyone who write Literature is a Writer”?

Part II
Queries

Chapter 4

Querying the semantic web

4.1 Motivation

We know how to express information on the web. It is now necessary to take advantage of it.

Nowadays, more resources are annotated via RDF due to its simple data model, formal semantics, and a sound and complete inference mechanism. RDF itself can be used as a query language for an RDF knowledge base using RDF consequence. Nonetheless, the use of consequence is still limited for answering queries. In particular, answering those that contain complex relations requires complex constructs. It is impossible, for example, to answer the query "find the names and addresses, if they exist, of persons who either work on query languages or ontology matching" using a simple consequence test.

Therefore the need for added expressivity in queries has led to define several query languages on top of graph patterns that are basically RDF and more precisely GRDF graphs. The focus of the next chapter is then to give an overview of some languages that have been designed or can be used for querying RDF graphs, and discusses the main differences between them in terms of expressiveness and limitations.

Example 11 (RDF). *RDF can be used for exposing a large variety of data. For instance, Figure 4.1 shows the RDF graph representing part of the gene regulation network acting in the drosophila embryo. Nodes represent genes and properties express regulation links, i.e., the fact that the expression of the source gene has an influence on the expression of the target gene. The triples of this graph are the following:*

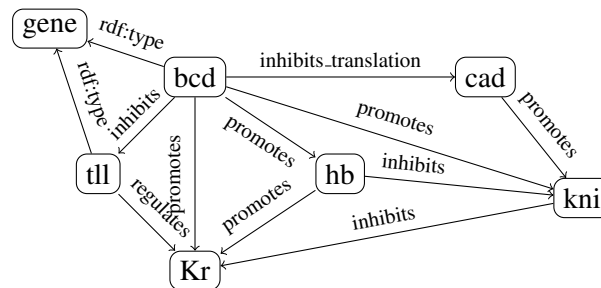


Figure 4.1: An RDF graph representing some interactions between genes within the early development of drosophila embryo.

```
dm:bcd rdf:type rn:gene.
dm:bcd rn:inhibits_translation dm:cad.
dm:bcd rn:promotes dm:hb.
dm:bcd rn:promotes dm:kni.
dm:bcd rn:promotes dm:Kr.
dm:bcd rn:inhibits dm:tll.
dm:cad rn:promotes dm:kni.
dm:hb rn:inhibits dm:kni.
dm:hb rn:promotes dm:Kr.
dm:kni rn:inhibits dm:Kr.
dm:tll rn:regulates dm:Kr.
dm:tll rdf:type rn:gene.
```

This example uses only URIs.

Example 12 (RDFS). *The RDF graph of Example 11 can be expressed in the context of an RDF Schema which provides more information about the vocabulary that it uses. It specifies subtypes of genes and subtypes of regulation relations.*

```
dm:maternal rdfs:subClassOf rn:gene.
dm:gap rdfs:subClassOf rn:gene.

dm:kni rdf:type dm:gap.
dm:hb rdf:type dm:gap.
dm:Kr rdf:type dm:gap.
dm:tll rdf:type dm:gap.
dm:bcd rdf:type dm:maternal.
dm:cad rdf:type dm:maternal.

rn:regulates rdfs:domain rn:gene.
rn:regulates rdfs:range rn:gene.
rn:inhibits rdfs:subPropertyOf rn:regulates.
rn:promotes rdfs:subPropertyOf rn:regulates.
rn:inhibits_translation rdfs:subPropertyOf rn:inhibits.
rn:inhibits_transcription rdfs:subPropertyOf rn:inhibits.
```

4.2 What is a query language?

Basically, the goal of a query language is to express queries that are evaluated against a representation. Query evaluation covers various aspects:

- Extracting information from a structure;
- Deducing information from a representation;
- Aggregating query answers;
- Constructing new structure.

4.2.1 Data extraction languages

The basic data extraction paradigm is the web. You know the URL, you got the HTML page. This is very basic and not very well structured: the operations for combining HTML pages are not very natural.

In the XML world, XPath is the appropriate language for extracting information. XML documents and answers are node sets to which queries can be applied and hence composed. XPath allows to navigate in the structure and offers powerful selection operations.

4.2.2 Data manipulation language

Data manipulation languages can do more than extraction languages. Following SQL, XQuery and XSLT typically provide XML as output. Not only they evaluate XPath queries for extracting information in XML documents, but they take advantage of the answers for constructing new XML documents. These documents can be fed again in XQuery or XSLT engines.

4.2.3 Query language as an algebra: SQL

A typical aspect of SQL is to be itself an algebra. I.e., queries are expression of an algebra that can be manipulated according to the rules of the algebra. This is used for distributing or rewriting queries for optimisation (with regard to a schema for instance).

4.2.4 Querying as testing consequences

So far, we did not consider deducting consequence from the representation: a query language is only here to extract and transform information.

Since we provided a semantics for semantic web languages, they should be evaluated with regard to this semantics. So instead of *extracting* information from a structure, the basic operation of a query language should be to know if information is consequence of the data. Since we provided different logics for expressing knowledge on the web, it is natural to be able to parameter queries with regard to these different logics (and the kind of entailment they offer). Hence, querying semantic web data can be expressed by:

$$\text{Data} + \text{Ontology} \models_L \phi$$

such that L is a particular representation language (Simple RDF, RDF, GRDF, RDF Schema, OWL Lite, OWL DL, OWL Full or others). In our case, the formula ϕ will be based on RDF graphs.

Chapter 5

Querying RDF with SPARQL

There has been early proposals for specific RDF query languages, such as RDQL [Seaborne, 2004], RQL [Karvounarakis *et al.*, 2002] or SeRQL [Broekstra, 2003]. In 2004, the W3C launched the Data Access Working Group for designing an RDF query language, called SPARQL, from these early attempts [Prud'hommeaux and Seaborne, 2008]. SPARQL query answering is characterized by defining maps from GRDF graphs used as query patterns of the query to the RDF knowledge base [Pérez *et al.*, 2006].

5.1 Syntax

5.1.1 SPARQL graph patterns

The heart of SPARQL queries is graph patterns. Informally, a *graph pattern* can be one of the following (see [Prud'hommeaux and Seaborne, 2008] for more details):

- a **triple pattern**: a triple pattern corresponds in RDF to a GRDF triple;
- a **basic graph pattern**: a set of triple patterns (or a GRDF graph) is called a basic graph patterns;
- a **union of graph patterns**: we use the keyword `UNION` in SPARQL to represent alternatives;
- an **optional graph pattern**: SPARQL allows optional results to be returned determined by the keyword `OPTIONAL`;
- a **constraint**: constraints in SPARQL are boolean-valued expressions that limit the number of answers to be returned. They can be defined using the keyword `FILTER`. As atomic `FILTER` expressions, SPARQL allows unary predicates like `BOUND`; binary (in)equality predicates (`=` and `!=`); comparison operators like `<`; data type conversion and string functions which will be omitted here. Complex `FILTER` expressions can be built using `!`, `||` and `&&`;
- a **group graph pattern**: is a graph pattern grouped inside `{` and `}`, and determines the scope of SPARQL constructs like `FILTER` and variable nodes;

Definition 23 (SPARQL graph pattern). A SPARQL graph pattern is defined inductively in the following way:

- every GRDF graph is a basic SPARQL graph pattern;

- if P_1 , P_2 are SPARQL graph patterns and K is a SPARQL constraint, then $\{P_1\}$, $(P_1 \text{ AND } P_2)$, $(P_1 \text{ UNION } P_2)$, $(P_1 \text{ OPTIONAL } P_2)$, and $(P_1 \text{ FILTER } K)$ are SPARQL graph patterns.

Example 13. The following graph pattern:

```
{ ?person foaf:knows "Faisal" . }
```

is a basic graph pattern that can be used in a query for finding persons who know Faisal.

```
{  
  { ?person ex:liveIn ex:France . }  
  UNION  
  { ?person ex:hasNationality ex:French . }  
}
```

is a union of two basic graph patterns that searches the persons who either live in France or have a French nationality.

The following graph pattern

```
{  
  ?person foaf:knows "Faisal" .  
  OPTIONAL  
  { ?person foaf:mbox ?mbox . }  
}
```

contains an optional basic graph pattern searching the mail boxes, if they exist, of persons who know Faisal.

```
{  
  ?person ex:liveIn ex:France .  
  ?person ex:hasAge ?age .  
  FILTER ( ?age < 40 ) .  
}
```

the constraint in this graph pattern limits the answers to the persons who live in France whose ages are less than 40.

```
{ { ?person foaf:knows "Faisal" . }  
  {  
    ?person ex:liveIn ex:France .  
    ?person ex:hasAge ?age .  
    FILTER ( ?age < 40 ) .  
  }  
}
```

is a graph pattern of two group graph patterns. The scope of the constraint in this graph pattern is the second group graph pattern. So, it is applied only to the persons who live in France.

5.1.2 SPARQL query

SPARQL provides several query forms.

Definition 24 (SPARQL query). Given a SPARQL graph pattern P , a tuple \vec{B} of variable in P , a URI u and a basic graph pattern Q ,

$$\begin{aligned} & \text{ASK FROM } u \text{ WHERE } P \\ & \text{SELECT } \vec{B} \text{ FROM } u \text{ WHERE } P \\ & \text{CONSTRUCT } Q \text{ FROM } u \text{ WHERE } P \end{aligned}$$

are SPARQL queries.

Intuitively, an answer to a SELECT query is an instantiation π of the variables of \vec{B} by the terms of the RDF graph G such that π is a restriction of a proof that P is a consequence of G . A CONSTRUCT query is used for building an RDF graph from the set of answers. ASK returns TRUE if there is an answer to a given query and FALSE otherwise. In addition, DESCRIBE is used for describing a resource RDF graph.

The following example queries give an insight of these query forms.

Example 14 (Query). *The following query searches, in the regulatory network of Figure 4.1, a gene $?x$ which inhibits a product that regulates a product that $?x$ promotes, and returns these three entities:*

```
SELECT ?x, ?y, ?z
FROM ...
WHERE
  ?x rn:inhibits ?y
  ?x rn:promotes ?z
  ?y rn:regulates ?z
  ?x rdf:type rn:gene.
```

Example 15. *The following ASK query:*

```
ASK
WHERE { ?person foaf:names "Faisal" .
        ?person ex:hasChild ?child .
}
```

returns TRUE if a person named Faisal has at least one child, FALSE otherwise.

The following CONSTRUCT query:

```
CONSTRUCT { ?son1 ex:brother ?son2 . }
WHERE {
  ?person foaf:names "Faisal" .
  ?son1 ex:sonOf ?person .
  ?son2 ex:sonOf ?person .
  FILTER ( ?son1 != ?son2 ) .
}
```

constructs the RDF graph (containing the brotherhood relation) by substituting for each located answer the values of the variables $?son1$ and $?son2$.

The following query:

```
DESCRIBE <example.org/person1>
```

returns a description of the resource identified by the given uriref, i.e., returns the set of triples involving this uriref.

SPARQL uses post-filtering clauses which allow, for example, to order (ORDER BY clause), or to limit (LIMIT and/or OFFSET clauses) the answers of a query. The reader is referred to the SPARQL specification [Prud'hommeaux and Seaborne, 2008] for more details or to [Pérez *et al.*, 2006] for formal semantics of SPARQL queries.

Example 16. *The following SPARQL query:*

```

SELECT ?name
WHERE {
  ?person ex:liveIn ex:France .
  ?person foaf:name ?name .
}
ORDER BY ?name
LIMIT 10
OFFSET 5

```

returns the names of persons who live in France limited to maximum 10 persons, ordered by their names, and starting from the 5th answer.

Since the graph patterns in the SPARQL query language are shared by all SPARQL query forms, we illustrate them using the SELECT ... FROM ... WHERE ... queries.

5.2 SPARQL Semantics

SPARQL query constructs are defined through algebraic operations on maps: assignments from a set of variables to terms that preserve names.

Definition 25 (Substitution or graph pattern map). *A substitution σ is a partial function from \mathcal{B} to \mathcal{T} . The domain of σ , denoted by $\text{dom}(\sigma)$, is the subset of \mathcal{B} on which σ is defined.*

Definition 26 (Application of a substitution to a basic graph pattern). *The application $\sigma(P)$ of a substitution σ to a basic graph pattern P , is defined by:*

- $\sigma(P) = \{\sigma(t); t \in P\}$ if P is a GRDF graph;
- $\sigma(\langle s, p, o \rangle) = \langle \sigma'(s), \sigma'(p), \sigma'(o) \rangle$ if P is a triple;
- $\sigma'(x) = \sigma(x)$ if $x \in \text{dom}(\sigma)$;

Operations on maps The restriction of σ to a set of terms X is defined by $\sigma|_X = \{\langle x, y \rangle \in \sigma \mid x \in X\}$ and the completion of σ to a set of terms X is defined by $\sigma^X = \sigma \cup \{\langle x, \text{null} \rangle \mid x \in X \text{ and } x \notin \text{dom}(\sigma)\}$.

If P is a graph pattern, then $\sigma(P)$ is the graph pattern obtained by the substitution of $\sigma(b)$ to each variable $b \in \mathcal{B}(P)$. Two maps σ_1 and σ_2 are *compatible* when $\forall x \in \text{dom}(\sigma_1) \cap \text{dom}(\sigma_2)$, $\sigma_1(x) = \sigma_2(x)$. Otherwise, they are said incompatible and this is denoted by $\sigma_1 \perp \sigma_2$. If σ_1 and σ_2 are two compatible maps, then we denote by $\sigma = \sigma_1 \oplus \sigma_2 : T_1 \cup T_2 \rightarrow \mathcal{T}$ the map defined by: $\forall x \in T_1, \sigma(x) = \sigma_1(x)$ and $\forall x \in T_2, \sigma(x) = \sigma_2(x)$.

In the following, we use an alternate characterization of SPARQL query answering that relies upon the correspondence between GRDF entailment and maps from the query graph patterns to the RDF graph [Pérez *et al.*, 2006]. The answers to a basic graph pattern query are those maps which warrant the entailment of the graph pattern by the queried graph. In the case of SPARQL, this entailment relation is GRDF entailment.

Definition 27 (Compound graph pattern entailment). *Let \models be an entailment relation on basic graph patterns, P, P' be SPARQL graph pattern, K be a SPARQL constraint, and G be an RDF*

graph, graph pattern entailment by an RDF graph modulo a map σ is defined inductively by:

$$\begin{aligned} G \models \sigma(P \text{ AND } P') &\text{ iff } G \models \sigma(P) \text{ and } G \models \sigma(P') \\ G \models \sigma(P \text{ UNION } P') &\text{ iff } G \models \sigma(P) \text{ or } G \models \sigma(P') \\ G \models \sigma(P \text{ OPTIONAL } P') &\text{ iff } G \models \sigma(P) \text{ and } [G \models \sigma(P') \\ &\text{ or } \forall \sigma'; G \models \sigma'(P'), \sigma \perp \sigma'] \\ G \models \sigma(P \text{ FILTER } K) &\text{ iff } G \models \sigma(P) \text{ and } \sigma(K) = \top \end{aligned}$$

The conditions K are interpreted as boolean functions from the terms they involve. Hence, $\sigma(K) = \top$ means that this function is evaluated to true once the variables in K are substituted by σ . If not all variables of K are bound, then $\sigma(K) \neq \top$.

The semantics of SPARQL FILTER expressions is defined as follows: given a map σ and a SPARQL constraint K , we say that σ satisfies K (denoted by $\sigma(K) = \top$), if:

- $K = \text{BOUND}(x)$ with $x \in \text{dom}(\sigma)$;
- $K = (x = c)$ with $x \in \text{dom}(\sigma)$ and $\sigma(x) = c$;
- $K = (x = y)$ with $x, y \in \text{dom}(\sigma)$ and $\sigma(x) = \sigma(y)$;
- $K = (x! = c)$ with $x \in \text{dom}(\sigma)$ and $\sigma(x)! = c$;
- $K = (x! = y)$ with $x, y \in \text{dom}(\sigma)$ and $\sigma(x)! = \sigma(y)$;
- $K = (x < c)$ with $x \in \text{dom}(\sigma)$ and $\sigma(x) < c$;
- $K = (x < y)$ with $x, y \in \text{dom}(\sigma)$ and $\sigma(x) < \sigma(y)$;
- $K = !K_1$ with $\sigma(K_1) = \perp$ (σ does not satisfy K_1);
- $K = (K_1 || K_2)$ with $\sigma(K_1) = \top$ or $\sigma(K_2) = \top$;
- $K = (K_1 \&\& K_2)$ with $\sigma(K_1) = \top$ and $\sigma(K_2) = \top$;

As usual for this kind of query language, an answer to a query is an assignment of distinguished variables (those variables in the SELECT part of the query). Such an assignment is a map from variables in the query to nodes of the graph. The defined answers may assign only one part of the variables, those sufficient to prove entailment. The answers are these assignments extended to all distinguished variables.

Definition 28 (Answer to a SPARQL query [Alkhateeb *et al.*, 2009]). *Let SELECT \vec{B} FROM u WHERE P be a SPARQL query with P a SPARQL graph pattern and G be the (G) RDF graph identified by the URI u , then the set of answers to this query is*

$$\mathcal{A}(\vec{B}, G, P) = \{\sigma|_{\vec{B}} \mid G \models_{GRDF} \sigma(P)\}.$$

This definition is a semantic characterization of SPARQL answers.

Example 17 (Query evaluation). *The evaluation of the query of Example 14 against the RDF graph of Example 11 returns only one answer:*

$$\langle \text{dm:bcd}, \text{dm:t11}, \text{dm:Kr} \rangle$$

In order to evaluate the complexity of query answering, we use the following problem, usually named query evaluation but better named ANSWER CHECKING:

Problem: \mathcal{A} -ANSWER CHECKING

Input: an RDF graph G , a SPARQL graph pattern P , a tuple of variables \vec{B} , and a map σ .

Question: Does $\sigma \in \mathcal{A}(\vec{B}, G, P)$?

This problem has usually the same complexity as checking if an answer exists. For SPARQL, the problem has been shown to be PSPACE-complete.

Proposition 6 (Complexity of \mathcal{A} -ANSWER CHECKING [Pérez *et al.*, 2009]). \mathcal{A} -ANSWER CHECKING is PSPACE-complete.

The complexity of checking RDF-entailment and GDRF-entailment is NP-complete [Gutierrez *et al.*, 2004]. This means that \mathcal{A} -ANSWER CHECKING when queries are reduced to basic graph patterns is NP-complete. In fact, the addition of AND, FILTER, and UNION does not increase complexity which remains NP-complete. This complexity comes from the OPTIONAL operation [Pérez *et al.*, 2009].

Hence, for every language in which entailment is NP-complete, used as a basic graph pattern language, the problem for this language will have the same complexity since Definition 27 shows the independence of subquery evaluation.

5.3 Algebraic manipulation

Answering SPARQL queries may be obtained by directly manipulating graphs and maps. The original semantics of SPARQL was given in this way. For defining these manipulations, we need some further definition coming from relational database theory. The *join* and *difference* of two sets of maps Ω_1 and Ω_2 are defined as follows [Pérez *et al.*, 2006]:

- (*join*) $\Omega_1 \bowtie \Omega_2 = \{\sigma_1 \oplus \sigma_2 \mid \sigma_1 \in \Omega_1, \sigma_2 \in \Omega_2 \text{ are compatible}\};$
- (*difference*) $\Omega_1 \setminus \Omega_2 = \{\sigma_1 \in \Omega_1 \mid \forall \sigma_2 \in \Omega_2, \sigma_1 \text{ and } \sigma_2 \text{ are not compatible}\}.$

Answers to compound graph patterns are obtained through the operations on maps.

Definition 29 (Answers to compound graph patterns). *Let P and P' be SPARQL graph patterns, K be a SPARQL constraint, and G be an RDF graph. The set $\mathcal{S}(P, G)$ of answers to P in G is defined inductively in the following way:*

$$\begin{aligned} \mathcal{S}(P, G) &= \{\pi|_{\mathcal{B}(P)} \mid \pi \text{ GRDF homomorphism from } P \text{ to } G\} \\ &\quad \text{if } P \text{ is a basic graph pattern} \\ \mathcal{S}((P \text{ AND } P'), G) &= \mathcal{S}(P, G) \bowtie \mathcal{S}(P', G) \\ \mathcal{S}(P \text{ UNION } P', G) &= \mathcal{S}(P, G) \cup \mathcal{S}(P', G) \\ \mathcal{S}(P \text{ OPTIONAL } P', G) &= (\mathcal{S}(P, G) \bowtie \mathcal{S}(P', G)) \cup (\mathcal{S}(P, G) \setminus \mathcal{S}(P', G)) \\ \mathcal{S}(P \text{ FILTER } K, G) &= \{\sigma \in \mathcal{S}(P, G) \mid \sigma(K) = \top\} \end{aligned}$$

The following corollary is obtained thanks to the interpolation lemma and its application to homomorphisms.

Corollary 7. *Let $Q = \text{SELECT } \vec{B} \text{ FROM } u \text{ WHERE } P$ be a SPARQL query, P be a GRDF graph and G be the (G) RDF graph identified by the URI u , then*

$$\mathcal{A}(\vec{B}, G, P) = \{\sigma|_{\vec{B}} \mid \sigma \in \mathcal{S}(P, G)\}.$$

Proof. By the interpolation lemma, if there exists an homomorphism from P to G , then $G \models_{GRDF} P$, moreover, this homomorphism determines an instance of G , hence $G \models_{GRDF} \pi(P)$. Hence, $\mathcal{S}(P, G) = \{\sigma|_{\mathcal{B}(P)} \mid G \models_{GRDF} \sigma(P)\}$, so $\{\sigma|_{\vec{B}} \mid \sigma \in \mathcal{S}(P, G)\} = \{\sigma|_{\vec{B}} \mid G \models_{GRDF} \sigma(P)\}$ which is exactly $\mathcal{A}(\vec{B}, G, P)$. \square

Then the corollary can be extended to any graph patterns showing the equivalence between the semantic definition (Definition 27) and the algebraic definition (Definition 29) by induction on the structure of graph patterns.

Proposition 8 (Answers to a SPARQL query). *Let $SELECT \vec{B} FROM u WHERE P$ be a SPARQL query, G be the RDF graph identified by the URI u , and $\mathcal{S}(P, G)$ be the set of answers to P in G , then the answers $\mathcal{A}(\vec{B}, G, P)$ to the query are the restriction and completion to \vec{B} of answers to P in G , i.e.,*

$$\mathcal{A}(\vec{B}, G, P) = \{\sigma|_{\vec{B}} \mid \sigma \in \mathcal{S}(P, G)\}.$$

This property is based on the fact that the answers to Q are the restrictions to \vec{B} of the set of RDF homomorphisms from P into G which corresponds to RDF-entailment.

5.4 Entailment regimes

The good news with the semantic definition is that SPARQL answers are defined semantically, independently of any evaluation mechanism. It is also defined with respect to a simple entailment relation. Hence it becomes possible to obtain a different query language by changing this entailment relation.

This is what is called an entailment regime and is currently under further development by the W3C [Glimm and Ogbuji, 2010] (however, still under the definition of “graph pattern homomorphisms” instead of a semantic definition). An entailment regime defines more than the change of entailment relation. It covers:

Graph language (G): which types of graphs are allowed (e.g., RDF, RDFS, OWL);

Query language (q): which kinds of queries are allowed (e.g., restriction or extension of the constraints);

Entailment relation (\models): the entailment relation used for defining the answers (e.g., RDF-entailment, RDFS-entailment, which OWL flavour);

Answer type (σ): the σ which will be taken as answer (e.g., are blanks allowed, skolemisation of blank nodes, does the solution lead to a consistent theory);

Syntax error handling : what has to be done in case q is defined incorrectly (e.g., raise an exception, ignore the incorrect part);

Inconsistency handling : what has to be done in case G is inconsistent (e.g., raising an exception, answering a particular subset of the answers).

We have defined SPARQL semantics so that it is easy to change the entailment regime. We show below how the query language can be changed (§6) and how the entailment relation may be changed for querying graphs modulo ontologies (§7).

5.5 Exercises

Exercise 8 (SPARQL queries). *Given the following SPARQL query:*

```

SELECT ?t
PREFIX
  foaf: http://xmlns.com/foaf/0.1/
  m: http://mydomain.com/myExample#
  dc: http://purl.org/dc/elements/1.1/
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
WHERE
  ?x foaf:name "Albert".
  ?x ?r ?l.
  ?r rdf:type rdf:Property.
  ?l rdf:type m:Roman.
  ?l dc:title ?t.

```

1. What is the informal meaning of this query?
2. Draw the RDF graph corresponding to the graph patterns of the query.
3. What is the difference between such graph patterns and simple RDF graphs?
4. Evaluate this query on each of the well-founded graphs of Figure 2.5 of Exercise 1 (p20) and provide the answer.
5. What must be added to this query for returning the year of publication of the *m:Roman* if it is available?

Exercise 9 (SPARQL and queries). Given the following SPARQL query:

```

SELECT ?m, ?n
PREFIX foaf: http://xmlns.com/foaf/0.1/
WHERE {
  ?x foaf:mbox ?m.
  ?x foaf:knows ?y.
  ?z foaf:mbox "keith@i.com".
  ?z marriedWith ?y.
  OPTIONAL { ?x foaf:name ?n }
}

```

1. What is, informally, the meaning of this query (tell it in English or French)?
2. Draw the GRDF graph corresponding to the graph pattern of this query.
3. Evaluate the query on the graphs of Figure 2.6 of Exercise 2 (p21) and provide the results.
4. Evaluate this query on the closure obtained at Question 4 of Exercise 5 and provide the results.

Exercise 10 (SPARQL query containment). Consider the following queries q_1 and q_2 on the RDF graph G of Exercise 3:

$$q_1 = \text{SELECT } ?w \text{ FROM } G \text{ WHERE } ((?w \text{ o:wrote } ?x) \text{ AND } \langle ?x \text{ rdf:type o:Poem} \rangle) \cup \langle ?w \text{ o:translated } ?x \rangle$$

$$q_2 = \text{SELECT } ?w \text{ FROM } G \text{ WHERE } ((?w \text{ o:wrote } ?l) \cup \langle ?w \text{ o:translated } ?l \rangle) \text{ AND } \langle ?l \text{ rdf:type o:Poem} \rangle$$

1. In the course, we defined the distinguished variables \vec{B} , the queried graph G and the query pattern P . Identify them in q_1 and q_2 .
2. Provide the answers of q_1 and q_2 with respect to the graph G .

Query containment $q \sqsubseteq q'$ between two queries $q = \text{SELECT } \vec{B} \text{ FROM } G \text{ WHERE } P$ and $q' = \text{SELECT } \vec{B} \text{ FROM } G \text{ WHERE } P'$ is defined by the fact that for any RDF graph, the answers to q are included in those to q' ($\forall G, \mathcal{A}(\vec{B}, G, P) \subseteq \mathcal{A}(\vec{B}, G, P')$).

3. What does the answer to the previous questions tell you about query containment between q_1 and q_2 ?
4. Do you think that query containment holds in some direction between q_1 and q_2 (either $q_1 \sqsubseteq q_2$ or $q_2 \sqsubseteq q_1$)?
5. Provide a proof for this. This may be done semantically by using the interpretation of query patterns or syntactically by translating queries into logic and showing that the query containment statement is a theorem.

Chapter 6

Extending SPARQL

SPARQL is a basic query language based on triple. However, for many applications, more elaborate used of queries are necessary. So, different work have attempted to improve the language in some directions.

There can be different approaches for extending, or reducing, SPARQL:

- adding operators (and, or, etc.), like `count (.)` or,
- changing the definition of basic graph patterns, like having path expressions instead of triples.

We will consider below several such extensions in the perspective of taking ontologies into account when querying data.

6.1 PPARQL

Both presented approaches have their drawbacks. Independently, we have designed a third approach which does not suffer from the same problems.

In order to address the problems raised by querying RDF graphs modulo RDF Schemas, we first present the PPARQL query language for RDF which we introduced independently in [Alkhateeb *et al.*, 2009]. Unlike SPARQL, PPARQL can express queries with regular path expressions instead of relations and variables in the edges. For instance, it allows for finding all maternal genes regulated by the `bcd` gene through an arbitrary sequence of inhibitions.

We will, in the next section, show how the additional expressive power provided by PPARQL can be used for answering queries modulo RDF Schema.

The added expressiveness of PPARQL is achieved by extending SPARQL graph patterns, hence any SPARQL query is a PPARQL query. SPARQL graph patterns, based on GRDF graphs, are replaced by PRDF graphs that are introduced below through their syntax (§6.1.1) and semantics (§6.1.2). They are then naturally replaced within the PPARQL context (§6.1.3).

6.1.1 PRDF syntax

Let Σ be an alphabet. A *language* over Σ is a subset of Σ^* : its elements are sequences of elements of Σ called *words*. A (non empty) word $\langle a_1, \dots, a_k \rangle$ is denoted $a_1 \cdot \dots \cdot a_k$. If $A = a_1 \cdot \dots \cdot a_k$ and $B = b_1 \cdot \dots \cdot b_q$ are two words over Σ , then $A \cdot B$ is the word over Σ defined by $A \cdot B = a_1 \cdot \dots \cdot a_k \cdot b_1 \cdot \dots \cdot b_q$.

Definition 30 (Regular expression pattern). *Let Σ be an alphabet, X be a set of variables, the set $\mathcal{R}(\Sigma, X)$ of regular expression patterns is inductively defined by:*

- $\forall a \in \Sigma, a \in \mathcal{R}(\Sigma, X)$ and $!a \in \mathcal{R}(\Sigma, X)$;
- $\forall x \in X, x \in \mathcal{R}(\Sigma, X)$;
- $\epsilon \in \mathcal{R}(\Sigma, X)$;
- If $A \in \mathcal{R}(\Sigma, X)$ and $B \in \mathcal{R}(\Sigma, X)$ then $A|B, A \cdot B, A^*, A^+ \in \mathcal{R}(\Sigma, X)$.

A regular expression over $(\mathcal{U}, \mathcal{B})$ can be used to define a language over the alphabet made of $\mathcal{U} \cup \mathcal{B}$. PRDF graphs are GRDF graphs where predicates in the triples are regular expression patterns constructed over the set of URI references and the set of variables.

Definition 31 (PRDF graph). *A PRDF triple is an element of $\mathcal{T} \times \mathcal{R}(\mathcal{U}, \mathcal{B}) \times \mathcal{T}$. A PRDF graph is a set of PRDF triples.*

Hence all GRDF graphs are PRDF graphs.

Example 18 (PRDF Graph). *PRDF graphs can express interesting features of regulatory networks. For instance, one may observe that $dm:bcd$ promotes $dm:Kr$ without knowing if this action is direct or indirect. Hence, this can be expressed by $dm:bcd \text{ rn:promotes+ } dm:Kr$. A generalized version of the graph pattern of the query of Example 14 can be expressed by:*

```
?x rn:inhibits.rn:regulates* ?z.
?x rn:promotes+ ?z.
?x rdf:type rn:gene.
```

6.1.2 PRDF semantics

To be able to define models of PRDF graphs, we have first to express path semantics within RDF semantics to support regular expressions.

Definition 32 (Support of a regular expression). *Let $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ be an interpretation of a vocabulary $V = \mathcal{U} \cup \mathcal{L}$, ι' be an extension of ι to $B \subseteq \mathcal{B}$, and $R \in \mathcal{R}(\mathcal{U}, B)$, a pair $\langle x, y \rangle$ of $(I_R \times I_R)$ supports R in ι' if and only if one of the two following conditions is satisfied:*

- (i) *the empty word $\epsilon \in L^*(R)$ and $x = y$;*
- (ii) *there exists a word of length $n \geq 1$ $w = w_1 \cdot \dots \cdot w_n$ where $w \in L^*(R)$ and a sequence of resources of I_R $x = r_0, \dots, r_n = y$ such that $\langle r_{i-1}, r_i \rangle \in I_{EXT}(\iota'(w_i))$, $1 \leq i \leq n$.*

Instead of considering paths in RDF graphs, this definition considers paths in the interpretations of PRDF graphs, i.e., paths are now relating resources. This is used in the following definition of PRDF models in which it replaces the direct correspondences that exist in RDF between a relation and its interpretation, by a correspondence between a regular expression and a sequence of relation interpretations. This allows for matching regular expressions, e.g., r^+ , with variable length paths.

Definition 33 (Model of a PRDF graph). *Let G be a PRDF graph, and $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ be an interpretation of a vocabulary $V \supseteq \mathcal{V}(G)$, I is a PRDF model of G if and only if there exists an extension ι' of ι to $\mathcal{B}(G)$ such that for every triple $\langle s, R, o \rangle \in G$, $\langle \iota'(s), \iota'(o) \rangle$ supports R in ι' .*

This definition extends the definition of RDF models, and they are equivalent when all regular expression patterns R are reduced to atomic terms, i.e., urirefs or variables. PRDF entailment is defined as usual:

Definition 34 (PRDF entailment). *Let P and G be two PRDF graphs, G PRDF-entails P (noted $G \models_{PRDF} P$) if and only if all models of G are models of P .*

6.1.3 PPARQL

PPARQL is an extension of SPARQL introducing the use of paths in SPARQL graph patterns. PPARQL graph patterns are built on top of PRDF in the same way that SPARQL is built on top of RDF.

Definition 35 (PPARQL graph pattern). *A PPARQL graph pattern is defined inductively by:*

- every PRDF graph is a PPARQL graph pattern;
- if P_1 and P_2 are two PPARQL graph patterns and K is a SPARQL constraint, then $(P_1 \text{ AND } P_2)$, $(P_1 \text{ UNION } P_2)$, $(P_1 \text{ OPT } P_2)$, and $(P_1 \text{ FILTER } K)$ are PPARQL graph patterns.

A PPARQL query for the select form is $\text{SELECT } \vec{B} \text{ FROM } u \text{ WHERE } P$ such that P is a PPARQL graph pattern.

Analogously to SPARQL, the set of answers to a PPARQL query is defined inductively from the set of maps of the PRDF graphs of the query into the RDF knowledge base. The definition of an answer to a PPARQL query will be thus identical to Definition 27, but it will use PRDF entailment.

Definition 36 (Answers to a PPARQL query). *Let $\text{SELECT } \vec{B} \text{ FROM } u \text{ WHERE } P$ be a PPARQL query with P a PRDF graph pattern, and G be the RDF graph identified by the URI u , then the set of answers to this query is:*

$$\mathcal{A}^*(\vec{B}, G, P) = \{\sigma|_{\vec{B}} | G \models_{PRDF} \sigma(P)\}$$

Chapter 7

Querying modulo ontologies

Our goal in this section is to consider how to answer SPARQL queries through taking into account the semantics of the RDF graph as a RDFS graph, i.e., by considering that answers are evaluated modulo an ontology. We will attempt at doing so by extending SPARQL. However,

[Muñoz *et al.*, 2007] introduces the reflexive relaxed semantics for RDFS in which `rdfs:subClassOf` and `rdfs:subPropertyOf` do not have to be reflexive. The entailment relation with this semantics is noted \models_{RDFS}^{nr} .

The reflexive relaxed semantics does not change much RDFS. Indeed, from the standard (reflexive) semantics, we can deduce that any class (respectively, property) is a subclass (respectively, subproperty) of itself. For instance, $\langle \text{dm:hb } \text{rn:inhibits } \text{dm:kni} \rangle$ only entails $\langle \text{rn:inhibits } \text{sp } \text{rn:inhibits} \rangle$ and variations of this triple in which occurrences of `rn:inhibits` are replaced by variables. The reflexivity requirement only entails reflexivity assertions which do not interact with other triples unless constraints are added to the `rdfs:subPropertyOf` or `rdfs:subClassOf` properties. We will call *genuine* an RDFS graph which does not constrain or extend the elements of the ρ DF vocabulary.

However, when issuing queries involving these relations, e.g., with a graph pattern like $\langle ?x \text{ sp } ?y \rangle$, all properties in the graph will be answers. Since this would clutter results, we assume, as done in [Muñoz *et al.*, 2009], that queries use the reflexive relaxed semantics. It is easy to recover the standard semantics by providing the additional triples when `sp` or `sc` are queried.

It is possible to define answers to SPARQL queries modulo RDF Schema, by using RDFS entailment as the entailment regime.

Definition 37 (Answers to a SPARQL query modulo RDF Schema). *Let $\text{SELECT } \vec{B} \text{ FROM } u \text{ WHERE } P$ be a SPARQL query with P a GRDF graph and G be the RDFS graph identified by the URI u , then the set of answers to this query modulo RDF Schema is:*

$$\mathcal{A}^\#(\vec{B}, G, P) = \{\sigma |_{\vec{B}} | G \models_{RDFS}^{nr} \sigma(P)\}$$

This definition is justified by the analogy between RDF entailment and RDFS entailment in the definition of answers to queries (see [Alkhateeb *et al.*, 2009]). It does not fully correspond to the RDFS entailment regime defined in [Glimm and Ogbuji, 2010] since we do not restrict the answer set to be finite. This is not necessary for the current treatment.

The problem is to specify a query engine that can find such answers. There are several ways to do this.

7.1 RDF(S) closure and query answering

One possible approach for querying an RDF(S) graph G in a sound and complete way is by computing the closure graph of G , i.e., the graph obtained by saturating G with all informations that can be deduced using a set of predefined rules called RDF(S) rules, then evaluating the query over the closure graph (see ter Horst closure in our Semantic web language lecture notes).

Example 19 (RDFS Closure). *The RDFS closure of the RDF graph of Example 11 augmented by the RDFS triples of Example 12 contains, in particular, the following assertions:*

```
dm:bcd rn:inhibits dm:cad. // [RDFS 9]
dm:hb rn:regulates dm:kni. // [RDFS 9]
dm:hb type rn:gene. // [RDFS 6]
```

Since queries must adopt the reflexive relaxed semantics, we have to further restrict this closure. It can be obtained by suppressing constraints RDFS8a and RDFS12a from the closure operation. We denote the partial non reflexive closure $\hat{G} \setminus H$.

Proposition 9 (Completeness of partial non reflexive RDFS closure). *Let G be a satisfiable genuine RDFS graph and H an RDFS graph, then $G \models_{RDFS}^{nrx} H$ if and only if $(\hat{G} \setminus H) \models_{RDF} H$.*

This entails the following corollary:

Corollary 10.

$$\mathcal{A}^\#(\vec{B}, G, P) = \mathcal{A}(\vec{B}, \hat{G} \setminus P, P)$$

This shows the correctness and completeness of the closure approach.

Example 20 (SPARQL evaluation modulo RDFS). *If the query of Example 14 is evaluated against the RDFS closure of Example 41, it will return the three expected answers:*

$$\begin{aligned} & \{y \langle dm:hb, dm:kni, dm:Kr \rangle \\ & \quad \langle dm:bcd, dm:tll, dm:Kr \rangle \\ & \quad \langle dm:bcd, dm:cad, dm:kni \rangle\} \end{aligned}$$

This can easily be obtained from the simple graph of Example 11 augmented by the triples of Example 41.

With this result, for the query evaluation problem, it is sufficient to enumerate the set of homomorphisms from the query graph pattern(s) into the closure graph.

This approach has several drawbacks which limited its use: It still tends to generate a very large graph which makes it not very convenient, especially if the transformation has to be made on the fly, i.e., when the query is evaluated; It takes time proportional to $|H| \times |G|^2$ in the worst case [Muñoz *et al.*, 2007]; Moreover, it is not applicable if one has no access to the graph to be queried but only to a SPARQL endpoint. In this case, it cannot compute the closure graph.

Since the complexity of the partial closure as been shown to be polynomial [ter Horst, 2005], $\mathcal{A}^\#$ -ANSWER CHECKING remains PSPACE-complete.

Proposition 11 (Complexity of $\mathcal{A}^\#$ -ANSWER CHECKING). *$\mathcal{A}^\#$ -ANSWER CHECKING is PSPACE-complete.*

7.2 The nSPARQL approach

Another possible approach [Muñoz *et al.*, 2007], consists of searching paths between RDF(S) vocabularies at query evaluation time. This results in a simple query language nSPARQL based on nested regular expressions for navigating the RDF graph. We offer here a slightly different use of these expressions that we name NSPARQL for differentiating it from the original nSPARQL [Pérez *et al.*, 2008]. However, the merits of the approach are directly inherited from the original nSPARQL. Our presentation differs on two points:

- We use a simplified, but equivalent, description of nested regular expressions;
- NSPARQL is simply SPARQL using nested regular expressions in predicate position of graph patterns. It is not fully clear that [Pérez *et al.*, 2008] considers exactly this.

Definition 38 (Nested regular expression). *A nested regular expression is an expression built from the following grammar:*

$$\text{exp} ::= \text{axis}\{\text{ }^{-1}\}\{ : : a \}\{ [\text{exp}] | \text{exp}/\text{exp} | \text{exp} | \text{exp} | \text{exp}^* \}$$

with $a \in \mathcal{U}$ and $\text{axis} \in \{\text{self}, \text{next}, \text{edge}, \text{node}\}$.

The model underlying nSPARQL is that of XPath which navigates within XML structures. Hence, axis denote the type of object which is selected at each step, respectively, the current node (self or self^{-1}), the nodes reachable through an outbound triple (next), the nodes that can reach the current node through an incident triple (next^{-1}), the predicates of outbound triples (edge), the predicates of incident triples (edge^{-1}), the object of a predicate (node) and the predicate of an object (node^{-1}). This is illustrated by Figure 7.1.

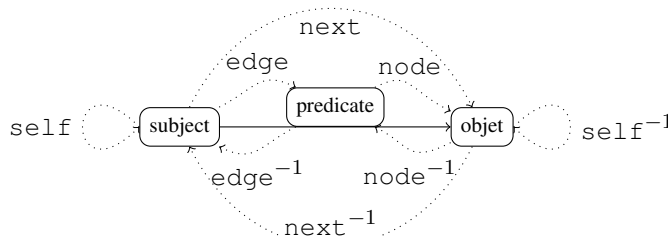


Figure 7.1: nSPARQL axis.

Example 21 (nSPARQL expressions). *In the context of molecular biology, nSPARQL expressions can be very useful. For instance, part of the graph patterns used for the query of Example 14 can be expressed by:*

```
?x next::rn:inhibits / next::rn:regulates ?z
```

which finds all pairs of nodes such that the first one inhibits a regulator of the second one. It can be further enhanced by using transitive closure:

```
?x next::rn:inhibits / next::rn:regulates+ ?z
```

expressing that we want a path between two nodes going through a first inhibition and then an arbitrary number of regulatory steps. Nested expressions allow for going further by constraining any step in the path. So,

```
?x next::rn:inhibits[ next::rdf:type/self::dm:gap ]
  / next::rn:regulates+ ?z
```

requires, in addition, the second node to be a gap gene.

Definition 39 (Basic nested path graph pattern). A basic nested path graph pattern is a triple $\langle s, p, o \rangle$ such that $s \in \mathcal{T}$, $o \in \mathcal{T}$ and p is a nested path.

A set of basic nested path graph patterns is called a nested path graph pattern. An NSPARQL query is a SPARQL query in which SPARQL graph patterns are replaced by nested path graph patterns, i.e., SPARQL graph patterns in which predicate are replaced by nested paths. Hence, NSPARQL graph patterns are built on top of nSPARQL in the same way that SPARQL is built on top of RDF.

Definition 40 (NSPARQL graph pattern). A NSPARQL graph pattern is defined inductively by:

- every nested path graph pattern is a NSPARQL graph pattern;
- if P_1 and P_2 are two NSPARQL graph patterns and K is a SPARQL constraint, then $(P_1 \text{ AND } P_2)$, $(P_1 \text{ UNION } P_2)$, $(P_1 \text{ OPT } P_2)$, and $(P_1 \text{ FILTER } K)$ are NSPARQL graph patterns.

In order to define answers to NSPARQL queries we need to know the semantics of nested paths. Here we depart from the semantics given in [Pérez *et al.*, 2008] by adding a third variable in the interpretation whose sole purpose is to compact the set of rules. Both definitions are equivalent.

Definition 41 (Nested path interpretation). Given a nested path p and an RDF graph G , the interpretation of p in G (denoted $\llbracket p \rrbracket_G$) is defined by:

$$\begin{aligned}
 \llbracket self \rrbracket_G &= \{ \langle x, x, x \rangle; x \in \mathcal{T} \} \\
 \llbracket next \rrbracket_G &= \{ \langle x, y, z \rangle; \exists z; \langle x, z, y \rangle \in G \} \\
 \llbracket edge \rrbracket_G &= \{ \langle x, z, z \rangle; \exists z; \langle x, z, y \rangle \in G \} \\
 \llbracket node \rrbracket_G &= \{ \langle z, y, z \rangle; \exists z; \langle x, z, y \rangle \in G \} \\
 \llbracket expr : : a \rrbracket_G &= \{ \langle x, y, a \rangle \in \llbracket expr \rrbracket_G \} \\
 \llbracket expr^{-1} \rrbracket_G &= \{ \langle y, x, z \rangle; \langle x, y, z \rangle \in \llbracket expr \rrbracket_G \} \\
 \llbracket exp_1 [exp_2] \rrbracket_G &= \{ \langle x, y, z \rangle \in \llbracket exp_1 \rrbracket_G; \\
 &\quad \exists w, k; \langle z, w, k \rangle \in \llbracket exp_2 \rrbracket_G \} \\
 \llbracket exp_1 / exp_2 \rrbracket_G &= \{ \langle x, y, z \rangle; \langle x, w, k \rangle \in \llbracket exp_1 \rrbracket_G \\
 &\quad \& \langle w, y, z \rangle \in \llbracket exp_2 \rrbracket_G \} \\
 \llbracket exp_1 \setminus exp_2 \rrbracket_G &= \llbracket exp_1 \rrbracket_G \cup \llbracket exp_2 \rrbracket_G \\
 \llbracket exp * \rrbracket_G &= \llbracket self \rrbracket_G \cup \llbracket exp \rrbracket_G \cup \llbracket exp / exp \rrbracket_G \\
 &\quad \cup \llbracket exp / exp / exp \rrbracket_G \cup \dots
 \end{aligned}$$

Definition 42 (Satisfaction of a basic nested path graph pattern). *Given a basic nested path graph pattern $\langle s, p, o \rangle$ and an RDF graph G , $\langle s, o \rangle$ satisfies p in G (denoted $G \models_{nSPARQL} \langle s, p, o \rangle$) if and only if there exists a map σ and $\exists \langle \sigma(s), \sigma(o), z \rangle \in \llbracket p \rrbracket_G$.*

Answers to NSPARQL queries follow the same definition as for Definition 27 using $\models_{nSPARQL}$ as the entailment relation.

Definition 43 (Answers to an NSPARQL query). *Let $SELECT \vec{B} FROM u WHERE P$ be a NSPARQL query with P a NSPARQL graph pattern and G be the (G)RDF graph identified by the URI u , then the set of answers to this query is:*

$$\mathcal{A}^o(\vec{B}, G, P) = \{\sigma \mid \vec{B} \mid G \models_{nSPARQL} \sigma(P)\}.$$

As presented in [Pérez *et al.*, 2008], nSPARQL can evaluate queries with regard to RDFS by transforming the queries with the following rules:

$$\begin{aligned} \phi(sc) &= (next::sc)+ \\ \phi(sp) &= (next::sp)+ \\ \phi(dom) &= next::dom \\ \phi(range) &= next::range \\ \phi(type) &= next::type/next::sc* \\ &\quad |edge/next::sp*/next::dom/next::sc* \\ &\quad |node^{-1}/next::sp*/next::range/next::sc* \\ \phi(p) &= next[(next::sp)*/self::p] \quad (p \notin \rho_{df}) \end{aligned}$$

This encoding is correct and complete with regard to entailment.

Proposition 12 (Completeness of ϕ [Pérez *et al.*, 2008] (Theorem 3)). *Given a NSPARQL graph pattern P and an RDFS graph G ,*

$$G \models_{RDFS}^{rx} P \text{ iff } G \models_{nSPARQL} \phi(P)$$

We can transfer this to NSPARQL query answering:

Proposition 13 (Answers to a SPARQL query modulo RDFS by NSPARQL). *Let $SELECT \vec{B} FROM u WHERE P$ be a SPARQL query with P a NSPARQL graph pattern and G the RDFS graph identified by the URI u , then the set of answers to this query is:*

$$\mathcal{A}^\#(\vec{B}, G, P) = \mathcal{A}^o(\vec{B}, G, \phi(P))$$

Example 22 (NSPARQL Query). *The result of transforming the query of Example 14 with ϕ is:*

```
SELECT ?x, ?y, ?z
FROM ...
WHERE
  ?x next[(next::sp)*/self::rn:inhibits] ?y.
  ?y next[(next::sp)*/self::rn:regulates] ?z.
  ?x next[(next::sp)*/self::rn:promotes] ?z.
  ?x next::rdf:type/next::sc*
    |edge/next::sp*/next::dom/next::sc*
    |node-1/next::sp*/next::range/next::sc* rn:gene.
```

This query provides the correct set of answers for the RDF graph of Example 11 modulo the RDF Schema of Example 12 (given in Example 20).

This approach gives a more efficient algorithm ($\mathcal{O}(n \log n)$ time complexity) for checking only the entailment between ground RDF(S) (or more precisely, restricted RDF(S)) graphs than using the closure operation. This algorithmic result also directly follows from the polynomial result of path satisfiability checking and the PRDF homomorphism [Alkhateeb, 2008] of ground graphs.

The main problem with NSPARQL is that, because nested regular expressions do not contain variables, it does not preserve SPARQL queries. Indeed, it is impossible to express a query containing the simple triple $\langle ?x ?y ?z \rangle$. This may seem like a minor problem, but in fact NSPARQL graph patterns prohibits dependencies between two variables as freely as it is possible to express in SPARQL.

7.3 Answering SPARQL queries modulo RDFS through PPARQL

To overcome the limitations of previous approaches to query RDF graphs modulo an RDF Schema, we provide a new approach which rewrites a SPARQL query into a PPARQL query using a set of rules, and then evaluates the transformed query over the graph to be queried. In particular, we show that every SPARQL query Q to evaluate over an RDFS graph G can be transformed into a PPARQL query $\tau(Q)$ such that evaluating Q over \hat{G} , the closure graph of G , is equivalent to evaluating $\tau(Q)$ over G .

The query rewriting approach is similar in spirit to the query rewriting methods using a set of views [Papakonstantinou and Vassalos, 1999; Calvanese *et al.*, 2000; Grahne and Thomo, 2003]. In contrast to these methods, our approach uses the data contained in the graph, i.e., the rules are inferred from RDFS entailment rules. We define a rewriting function τ from RDF graph patterns to PRDF graph patterns through a set of rewriting rules over triples (which naturally extends to basic graph patterns and queries). $\tau(Q)$ is obtained from Q by applying the possible rule(s) to each triple in Q .

Definition 44 (Basic RDFS graph pattern expansion). *Given an RDF triple t , the RDFS expansion of t is a finite PPARQL graph pattern $\tau(t)$ defined as:*

$$\begin{aligned} \tau(\langle s, sc, o \rangle) &= \{ \langle s, sc^+, o \rangle \} \\ \tau(\langle s, sp, o \rangle) &= \{ \langle s, sp^+, o \rangle \} \\ \tau(\langle s, p, o \rangle) &= \{ \langle s, ?x, o \rangle, \langle ?x, sp^*, p \rangle \} (p \notin \{sc, sp, type\}) \\ \tau(\langle s, type, o \rangle) &= \{ \langle s, type \cdot sc^*, o \rangle \} \\ &\quad \cup \{ \langle s, ?p, ?y \rangle, \langle ?p, sp^*, ?q \rangle, \langle ?q, dom \cdot sc^*, o \rangle \} \\ &\quad \cup \{ \langle ?y, ?p, s \rangle, \langle ?p, sp^*, ?q \rangle, \langle ?q, range \cdot sc^*, o \rangle \} \end{aligned}$$

The first rule handles the transitive semantics of the subclass relation. Finding the subclasses of a given class can be achieved by navigating all its direct subclasses. The second rule handles similarly the transitive semantics of the subproperty relation. The third rule tells that the subject-object pairs occurring in the subproperties of a given property are inherited to it. Finally, the fourth rule expresses that the instance mapped to s has for type the class mapped to o (we use the word "mapped" since s and/or o can be variables) if one of the following conditions holds:

1. the instance mapped to s has for type one of the subclasses of the class mapped to o by following the subclass relationship zero or several times. The zero times is used since s can be directly of type o ;
2. there exists a property of which s is subject and such that the instances appearing as a subject must have for type one of the subclasses of the class mapped to o ;
3. there exists a property of which s is object and such that the instances appearing as an object must have for type one of the subclasses of the class mapped to o .

The latter rule takes advantage of a feature of PSPARQL which does not exist in NSPARQL: the ability to have variables in predicates.

Example 23 (PSPARQL Query). *The result of transforming the query of Example 14 with τ is:*

```
SELECT ?x, ?y, ?z
FROM ...
WHERE
  ?x ?r ?y. ?r sp* rn:inhibits.
  ?y ?t ?z. ?t sp* rn:regulates.
  ?x ?s ?z. ?s sp* rn:promotes.
  ( ?x rdf:type.sc* rn:gene.
  UNION
    { ?x ?u ?v. ?u sp*.dom.sc* rn:gene.}
  UNION
    { ?v ?u ?x. ?u sp*.range.sc* rn:gene.}
  )
```

This query provides the correct set of answers for the RDF graph of Example 11 modulo the RDF Schema of Example 12 (given in Example 20).

Now we can show that any SPARQL query can be answered modulo an RDF Schema by rewriting the resulting query in PSPARQL and evaluating the PSPARQL query.

Proposition 14 (Answers to a SPARQL query modulo RDF Schema by PSPARQL).

$$\mathcal{A}^\#(\vec{B}, G, P) = \mathcal{A}^*(\vec{B}, G, \tau(P))$$

This transformation does not increase the complexity of PSPARQL which is the same as the one of SPARQL:

Proposition 15 (Complexity of \mathcal{A}^* -ANSWER CHECKING). \mathcal{A}^* -ANSWER CHECKING is PSPACE-complete.

7.4 Querying modulo DL-lite

DL-Lite is a restriction of the OWL language (see Languages of the semantic web, [Calvanese *et al.*, 2007]). It allows for efficient query answering through rewriting the query.

In this case, queries are restricted to disjunctions of conjunctive atoms. This means the SPARQL queries must be UNION of basic graph patterns.

The result is obtained by using the DL-Lite terminology in order to transform the query into a set (union) of transformed queries that can be evaluated independently. It also requires that the ontology be satisfiable.

See Marie-Christine Rousset slides for more details.

7.5 Conclusion

The approaches presented here are specific to SPARQL and RDF schema. These cannot be easily extended to more expressive languages like languages of the OWL family. Concerning RDFS queries, research is very active to find the best way to evaluate these queries.

7.6 Exercises

Exercise 11 (SPARQL modulo ontologies). *1. The way queries are answered in Exercise 9 is not the standard way for answering queries modulo DL-Lite ontologies. What is the standard method? Rewrite the query of Exercise 9 with this method and give its results.*

Exercise 12 (Query modulo ontology). *We now consider the ontology o of Exercise 7 and the following queries:*

- $q_3 = \text{SELECT } ?y \text{ FROM } o \text{ WHERE } \langle ?x, o:\text{translated}, ?y \rangle;$
- $q_4 = \text{SELECT } ?y \text{ FROM } o \text{ WHERE } \langle ?y, rdf:\text{type}, o:\text{Literature} \rangle.$

1. *Do you think that query containment holds in some direction between q_3 and q_4 (either $q_3 \sqsubseteq q_4$ or $q_4 \sqsubseteq q_3$)? Tell why.*
2. *Can you provide a definition for query containment modulo an ontology o ($q \sqsubseteq_o q'$)?*
3. *Does it return different answers for q_3 and q_4 ? (either $q_3 \sqsubseteq_o q_4$ or $q_4 \sqsubseteq_o q_3$)? Tell why.*

Part III

Networks of ontologies

Chapter 8

Networks of ontologies and alignments

8.1 Motivation

We are dealing with a space of knowledge sources called the semantic web. Each source provides knowledge expressed in its own vocabulary called ontology¹. There is no a priori reason that this knowledge be expressed:

- in the same knowledge representation language,
- with the same vocabulary, or
- with the same modelling of that vocabulary.

However, this heterogeneity should not prevent from interpreting this knowledge, like the fact that people use different natural languages and may have different state of mind does not fully prevent them to understand each others.

Example 24 (Heterogeneity). *Consider two knowledge sources to be found on the the web The first one, o , is a simple set of triples :*

```
May childOf Peter.  
Peter childOf Paul.  
Paul childOf Zeno.
```

The second source, o' , is an axiom in some first order logic dialect:

```
grandParent( x, y )  $\Leftarrow$   $\exists$  z; parent( x, z )  $\wedge$  parent( z, y )
```

One would expect, given these two sources to be able to deduce that $\text{grandParent}(\text{Paul}, \text{May})$ (that we will refer to as δ holds. This is however not the case since:

- *The two sources use different knowledge representation languages (syntactic heterogeneity);*
- *The two sources use different vocabularies: child versus parent (terminological heterogeneity);*
- *Even if the vocabulary were the same, the modelling of these notion may be different.*

This can be summarised by the following remarks:

- $o \not\models \delta$ and $o' \not\models \delta$;

¹In fact, an ontology provides both the vocabulary and axioms governing the terms in the vocabulary.

- $o \cup o'$ impossible
- even $o \cup o' \not\models \delta$

This problem can be solved by imposing to the word a unique ontology and a unique knowledge representation language. However, this is both unrealistic and unwishable. Instead, we propose to draw relations between knowledge sources mostly based on their ontologies. We call “ontology matching” the process of finding the relations between ontologies and we call “alignment” the result of this process expressing declaratively these relations. In an open world in which ontologies evolve, managing ontologies requires using alignments for expressing the relations between ontologies.

Example 25 (Alignment). *In the present case, we can imagine to provide an alignment A containing the following correspondence:*

$$\langle \text{childOf}^{-1}, \equiv, \text{parent} \rangle$$

this alignment would solve our current problem because $o \cup o' \cup A \models \delta$. However, the merge of the ontologies provided by $o \cup o' \cup A$ may not be the solution that works in any case. For instance, if o' contained in addition, the following axioms:

```
parent( x, y )  $\Rightarrow$  Human( x )  $\wedge$  Human( y )
childOf( x, y )  $\Rightarrow$  Tree( x )  $\wedge$  Tree( y )
Human  $\perp$  Tree
```

*The merge would be an inconsistent knowledge source. In such a case, it may be better to use the alignment as a data transformation programme that would import whatever *childOf* formula entailed by o into the corresponding *parent* formula in o' .*

Several classes of applications can be considered (they are more extensively described in [Euzenat *et al.*, 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 [Rousset *et al.*, 2006], Figure 8.1 is an example of the use of alignments for mediation in such systems;

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;

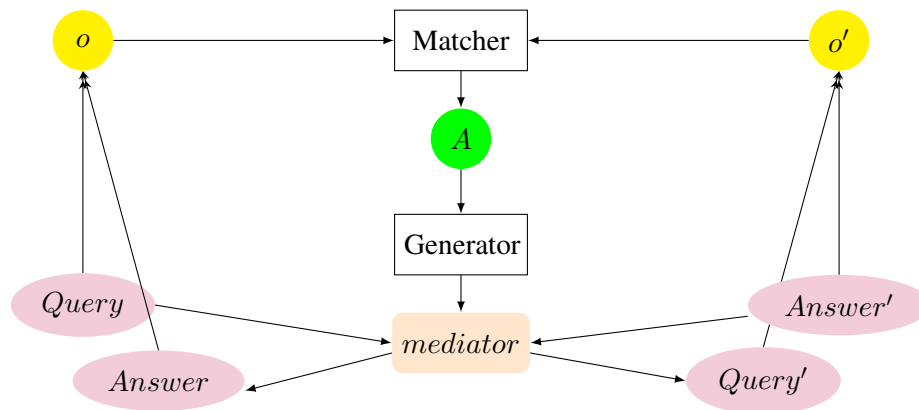


Figure 8.1: Query mediation (from [Euzenat *et al.*, 2007]). From two matched ontologies o and o' , resulting in alignment A , a *mediator* is generated. This allows the transformation of queries expressed with the entities of the first ontology into a query using the corresponding entities of a matched ontology and the translation back of the results from the second ontology to the first one.

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 about 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 area of ontology, but applies to any application that communicates through ontologies.

We consider here, the foundations of ontology matching and alignments within the semantic web. In particular, we will establish the semantics of alignments and the constraints this raises on the interpretation of ontologies.

We first precisely define what are alignments through their syntax (§9) and semantics (§11) before introducing networks of ontologies and their semantics. There is no “standard” semantics for networks of ontologies, so we provide here an abstract view that aims at covering all of them and we provide an example with our equalising semantics (§12). Finally, we consider the use of the semantics of networks of ontologies within the context of semantic peer-to-peer systems (§13.1).

8.2 Reminder: ontology semantics

We consider ontologies as logical theories. The languages used in the semantics web such as RDF or OWL are indeed logics. An ontology is a set of assertions that selects the set of interpretations which satisfy them. These interpretations are called models. They constitute the possible interpretations of an ontology.

Definition 45 (Model). *Given an ontology o , a model of o is an interpretation m of o , which satisfies all the assertions in o :*

$$\forall \delta \in o, m \models \delta$$

The set of models of an ontology o is denoted as $\mathcal{M}(o)$.

Finally, an important notion is the set of assertions that are consequences of an ontology. These are the assertions implicitly entailed by an ontology and they determine the answers to queries.

Definition 46 (Consequence). *Given an ontology formula δ , δ is a consequence of an ontology o , if and only if, it is satisfied by all models of o . This is denoted as $o \models \delta$.*

Given a model m , we will denote as $m(e)$ the application of the interpretation function of the model to some ontology entity e .

This digression introduced more precisely, albeit generally, a simplified syntax and semantics of ontologies. This will be useful when considering the meaning of matching ontologies.

Chapter 9

Alignment syntax and networks of ontologies

Alignments express the correspondences between entities belonging to different ontologies¹. All definitions here are given for matching between two ontologies. In case of multiple matching, the definitions can be straightforwardly extended by using n -ary correspondences. A correspondence must consider the two corresponding entities and the relation that is supposed to hold between them. We provide the definition of the alignment following the work in [Euzenat, 2004; Bouquet *et al.*, 2004].

Since the related entities are an important part of alignments, they have to be defined. We separate the matched entities from the ontology language because it can be desirable to have a different language for identifying the matched entities. Given an ontology language, we use an *entity language* for expressing those entities that will be put in correspondence by matching. The expressions of this language will depend on the ontology on which expressions are defined.

The entity language can be simply made of all the formulas of the ontology language based on the ontology vocabulary. It can restrict its scope to particular kinds of formulas from the language, for instance, atomic formulas, or even to terms of the language, like class expressions. It can also restrict the entities to be only named entities. This is convenient in the context of the semantic web to restrict entities to those identifiable by their URIs. The entity language can also be an extension of the ontology language: this can be a query language, such as SPARQL [Prud'hommeaux and Seaborne, 2008], adding operations for manipulating ontology entities that are not available in the ontology language itself, like concatenating strings or joining relations. Finally, this entity language can combine both extension and restriction, e.g., by authorising any boolean operations over named ontology entities.

Definition 47 (Entity language). *Given an ontology language L , an entity language Q_L is a function from any ontology $o \subseteq L$ which defines the matchable entities of ontology o .*

In the following we will assume that each ontology interpretation can be extended to an interpretation of the entity language associated with the ontology.

The next important component of the alignment is the relation that holds between the entities. We identify a set of relations Θ that is used for expressing the relations between the entities. Matching algorithms primarily use the equivalence relation (\equiv) meaning that the matched objects are the same or are equivalent if these are formulas. It is possible to use relations from

¹This part is mostly written from [Euzenat *et al.*, 2007].

the ontology language within Θ . For instance, using OWL, it is possible to take advantage of the `owl:equivalentClass`, `owl:disjointWith` or `rdfs:subClassOf` relations in order to relate classes of two ontologies. These relations correspond to set-theoretic relations between classes: *equivalence* ($=$); *disjointness* (\perp); *more general* (\sqsupseteq). They can be used without reference to any ontology language. Finally, relations can be of any type and are not restricted to relations present within the ontology language, such as fuzzy relations or probability distributions over a complete set of relations, or similarity measures.

A more general view is the introduction of algebras of relations instead of these ad hoc sets of relations [Euzenat, 2008].

With these ingredients, it is possible to define the correspondences that have to be found by matching algorithms.

Definition 48 (Correspondence). *Given two ontologies o and o' with associated entity languages Q_L and $Q_{L'}$ and a set of alignment relations Θ , a correspondence is a triple:*

$$\langle e, e', r \rangle,$$

such that

- $e \in Q_L(o)$ and $e' \in Q_{L'}(o')$;
- $r \in \Theta$.

The correspondence $\langle e, e', r \rangle$ asserts that the relation r holds between the ontology entities e and e' .

Example 26 (Correspondence). *For example, a simple kind of correspondence is as follows:*

`http://book.ontologymatching.org/example/culture-shop.owl#Book =`
`http://book.ontologymatching.org/example/library.owl#Volume`

It asserts the equivalence relation with full confidence between what is denoted by two URIs, namely the `Book` class in one ontology and the `Volume` class in another one. Some examples of more complex correspondences are as follows:

$$\begin{aligned} & \text{author}(x, \text{concat}(w.\text{firstname}, w.\text{lastname})) \Leftarrow_{.85} \wedge \text{writtenBy}(x, w) \\ & \wedge \text{Writer}(w) \\ & \text{Book}(x) \end{aligned}$$

is a Horn clause expressing that if there exists a `Book` x written by `Writer` w , the author of x in the first ontology is identified by the concatenation of the first and last name of w . The confidence in this clause is quantified with a .85 degree.

There can be several possible correspondences for the same entities depending on the language in which correspondences are expressed. For instance, one could have the simple correspondence that `speed` in one ontology is equivalent to `velocity` in another one:

$$\text{speed} \equiv \text{velocity}$$

or record that they are expressed in miles per hour and metre per second respectively:

$$\begin{aligned} \text{speed} &= \text{velocity} \times 2.237 \\ 0.447 \times \text{speed} &= \text{velocity} \end{aligned}$$

For pragmatic reasons, the relationship between two entities is often assigned a degree of confidence which can be viewed as a measure of trust in the fact that the correspondence holds – ‘I trust 70% the fact that the correspondence is correct or reliable’ – and can be compared with the certainty measures provided with meteorological forecasts. This measure is taken from a confidence structure.

Definition 49 (Confidence structure). *A confidence structure is an ordered set of degrees $\langle \Xi, \leq \rangle$ for which there exists a greatest element \top and a smallest element \perp .*

In [Euzenat *et al.*, 2007], confidence was included in Definition 48 (then Definition 2.11) as an additional component. We now consider that it is better to consider confidence as metadata over the correspondence itself.

The usage of confidence degrees is that the higher the degree with regard to \leq , the most likely the relation holds. It is convenient to interpret the greatest element as the boolean true and the smallest element as the boolean false.

The most widely used structure is based on the real number unit interval $[0, 1]$, but some systems simply use the boolean lattice. Some other possible structures are fuzzy degrees, probabilities or other lattices. [Gal *et al.*, 2005] has investigated the structure of fuzzy confidence relations. This structure can be extended, for instance, if one wants to compose alignments. Thus, in this case, it may be necessary to define operations for combining these degrees.

Finally, an alignment is defined as a set of correspondences.

Definition 50 (Alignment). *Given two ontologies o and o' , an alignment is a set of correspondences between pairs of entities belonging to $Q_L(o)$ and $Q_L(o')$ respectively.*

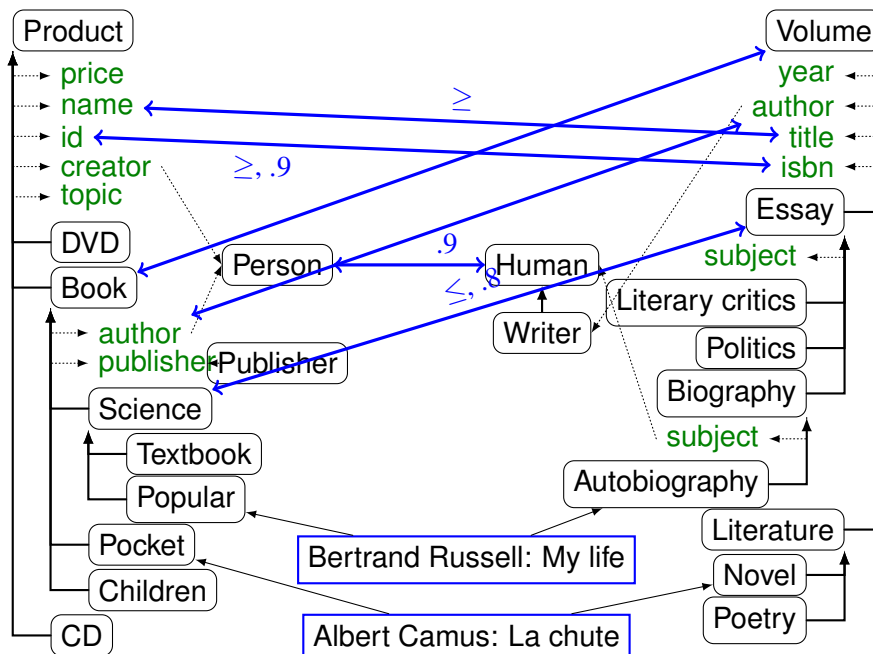


Figure 9.1: Alignment between two ontologies. Correspondences are expressed by arrows. By default their relation is = and their confidence value is 1.0; otherwise, these are mentioned near the arrows.

Example 27 (Alignment). *Figure 9.1 displays a possible alignment for a pair of ontologies. It can be expressed by the following correspondences:*

$$\begin{array}{ll} \text{Book} =_{1.0} \text{Volume} & \text{name} \geq_{1.0} \text{title} \\ \text{id} \geq_{.9} \text{isbn} & \text{author} =_{1.0} \text{author} \\ \text{Person} =_{.9} \text{Human} & \text{Science} \leq_{.8} \text{Essay} \end{array}$$

In the following, for the sake of simplicity, we only consider that ontologies are description logic terminologies and their entities are defined classes identified by URIs. We will only consider $=$, \leq , \geq and \perp as relations expressing equivalence, subsumption and disjointness between classes. However, results will also apply to other relations.

So far, our alignments are very simple: they are sets of pairs of entities from two ontologies. However, there are, in the literature, at least three types of n:m, multiple or complex alignments:

1. alignments involving more than two ontologies produced by multiple matching, that we may call multialignments,
2. alignments involving correspondences between more than two entities (still belonging to two ontologies),
3. alignments with entities involved in more than one correspondence that are denoted by the use of * (zero-or-more) or + (more-than-zero) in their cardinalities.

In case of multiple matching (1), the alignments must contain correspondences relating more than two entities. The definitions above must then be extended accordingly. This is not covered further here.

The second kind of correspondences (2) can be thought of as using non binary relations. For instance, in schema matching, some authors [Sheth and Larson, 1990; Rahm and Bernstein, 2001] tend to consider that a correspondence like:

$$\text{address} = \text{street} + \text{“ ”} + \text{number}$$

is a ternary complex relation ($\cdot = \cdot + \text{“ ”} + \cdot$) between three entities address, street and number. However, given the nature of the problem: matching two ontologies, we will consider that these objects can be grouped by operators in the entity language Q_L which may include operators such as concatenation, arithmetic operations or logical connectors for that purpose. Hence, in our setting, the correspondence above is simply a normal correspondence in which the binary relation is equivalence ($=$) and the ontology entities are address and $\text{street} + \text{“ ”} + \text{number}$. This is the main reason why we consider ontology entities, the latter entity is a term built on strings and operations on strings (here concatenation $+$). In its simplest expression the only construction can be a set.

Option (3) is related to the multiplicity of the alignment if it is considered as a relation. By analogy with mathematical functions, it is useful to define some properties of the alignments. These apply when the only considered relation is equality ($=$) and the confidence measures are not taken into account. One can ask for a total alignment with regard to one ontology, i.e., all the entities of one ontology must be successfully mapped to the other one. This property is purposeful whenever thoroughly transcribing knowledge from one ontology to another is the goal: there is no entity that cannot be translated.

One can also require the alignment to be injective with regard to one ontology, i.e., all the entities of the other ontology is part of at most one correspondence. Injectivity is useful in ensuring that entities that are distinct in one ontology remain distinct in the other one. In particular, this contributes to the reversibility of alignments.

Definition 51 (Total alignment, injective alignment). *Given two ontologies o and o' , an alignment A over o and o' is called a total alignment from o to o' if and only if:*

$$\forall e \in Q_L(o), \exists e' \in Q_{L'}(o'); \langle e, e', = \rangle \in A$$

and, it is called an injective alignment from o to o' if and only if:

$$\forall e' \in Q_{L'}(o'), \exists e_1, e_2 \in Q_L(o); \langle e_1, e', = \rangle \in A \wedge \langle e_2, e', = \rangle \in A \implies e_1 = e_2$$

These properties heavily depend on the ontology entity languages which are chosen for alignments.

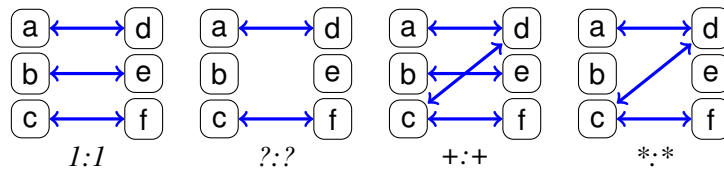
Usual mathematical properties apply to these alignments. In particular, a total alignment from o to o' is a surjective alignment from o' to o . A total alignment from both o and o' , which is injective from one of them, is a bijection. In mathematical English, an injective function is said to be *one-to-one* and a surjective function to be *onto*. Due to the wide use among matching practitioners of the term *one-to-one* for a bijective, i.e., both injective and surjective, alignment, we will only use one-to-one for bijective.

Finally, we can extend these definitions when correspondence relations are *not* equivalence. In such a case, they do not ensure the same properties. For instance, injectivity does not guarantee reversibility of the alignment used as a transformation.

In conceptual models and databases, the term multiplicity denotes the constraints on a relation. Usual notations are 1:1 (one-to-one), 1:m (one-to-many), n:1 (many-to-one) or n:m (many-to-many). If we consider only total and injective properties, denoted as 1 for injective and total, ? for injective, + for total and * for none, and the two possible orientations of the alignments, from o to o' and from o' to o , the multiplicities become: ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:* , *:?, 1:* , *:1, +:* , *:+, *:* [Euzenat, 2003].

Example 28 (Alignment multiplicity). *The alignment of Table 27 is ?:?. If we add the correspondence $\text{Product} \geq \text{Volume}$, then it is ?:* . If we now consider relating any entity of the second ontology to another entity of the first one, then it becomes ?:+.*

The four pictures below display some of the possible configurations for two ontologies composed of three classes each.



9.1 Networks of ontologies

These definitions can be generalised to an arbitrary number of alignments and ontologies captured in the concept of a network of ontologies (or distributed system in the sense of [Ghidini and Serafini, 1998; Franconi *et al.*, 2003]), i.e., sets of ontologies and alignments.

Definition 52 (Network of ontologies). *A network of ontologies $\langle \Omega, \Lambda \rangle$ is made of a set Ω of ontologies and a set Λ of alignments between these ontologies. We denote as $\Lambda(o, o')$ the set of alignments in Λ between o and o' .*

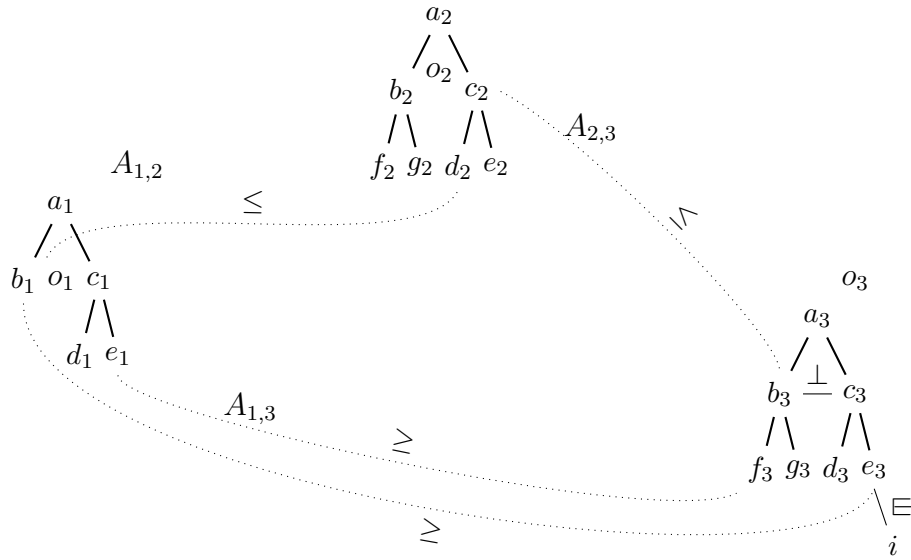


Figure 9.2: A network of ontologies made of three ontologies (o_1 , o_2 , and o_3) and three alignments ($A_{1,2}$, $A_{1,3}$, and $A_{2,3}$).

Example 29 (Network of ontologies). *Figure 9.2 presents three ontologies:*

$$\begin{aligned}
 o_1 &= \left\{ \begin{array}{l} b_1 \sqsubseteq a_1 \quad c_1 \sqsubseteq a_1 \\ d_1 \sqsubseteq c_1 \quad e_1 \sqsubseteq c_1 \end{array} \right. \\
 o_2 &= \left\{ \begin{array}{l} b_2 \sqsubseteq a_2 \quad c_2 \sqsubseteq a_2 \quad g_2 \sqsubseteq b_2 \\ d_2 \sqsubseteq c_2 \quad e_2 \sqsubseteq c_2 \quad f_2 \sqsubseteq b_2 \end{array} \right. \\
 o_3 &= \left\{ \begin{array}{l} b_3 \sqsubseteq a_3 \quad c_3 \sqsubseteq a_3 \quad g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3 \quad e_3 \sqsubseteq c_3 \quad f_3 \sqsubseteq b_3 \\ i \sqsubseteq e_3 \quad b_3 \perp c_3 \end{array} \right.
 \end{aligned}$$

together with three alignments $A_{1,2}$, $A_{2,3}$, and $A_{3,1}$. These alignments can be described as follows:

$$\begin{aligned}
 A_{1,2} &= \{ b_1 \leq d_2 \} \\
 A_{2,3} &= \{ c_2 \leq b_3 \} \\
 A_{1,3} &= \{ e_1 \geq f_3 \quad b_1 \geq e_3 \}
 \end{aligned}$$

Definition 53 (Normalised network of ontologies). *A network of ontologies $\langle \Omega, \Lambda \rangle$ is said normalised if and only if for any two ontologies o and o' , $|\Lambda(o, o')| = 1$.*

In a normalised ontology, we denote $\lambda(o, o')$ the unique alignment between o and o' .

Any network of ontologies may be easily normalised by:

- if $|\Lambda(o, o')| = 0$, adding an empty alignment between o and o' ,
- if $|\Lambda(o, o')| > 1$, replacing $\Lambda(o, o')$ by a unique alignment containing all the correspondences of the alignments of $\Lambda(o, o')$,

There are better ways to normalise such networks, but this simple one is sufficient for obtaining equivalent normalised networks. Hence, from now we will only consider normalised networks of ontologies.

We can define syntactic subsumption between networks of ontologies. This definition is given between unrestricted networks, but it is really useful on normalised networks.

Definition 54 (Syntactic subsumption between networks of ontologies). *Given two networks of ontologies, $\langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$, $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$ iff $\exists \langle h, k \rangle$, a pair of morphisms: $h : \Omega \longrightarrow \Omega'$ and $k : \Lambda \longrightarrow \Lambda'$ such that $\forall o \in \Omega, \exists h(o) \in \Omega'$ and $o \subseteq h(o)$ and $\forall A \in \Lambda(o, o'), \exists k(A) \in \Lambda'(h(o), h(o'))$ and $A \subseteq k(A)$.*

This means that a network of ontology is syntactically subsumed by another network if any ontology (respectively any alignment) of the former has a counterpart in the latter one which contains at least all of its axioms (respectively correspondences) and that the graph structure of the former network is preserved in the latter. It is possible that several ontologies have the same counterpart as soon as these conditions are met.

We note: $\langle \Omega, \Lambda \rangle \equiv \langle \Omega', \Lambda' \rangle$ iff $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$ and $\langle \Omega, \Lambda \rangle \supseteq \langle \Omega', \Lambda' \rangle$; $\langle \Omega, \Lambda \rangle \sqsubset \langle \Omega', \Lambda' \rangle$ iff $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$ and $\langle \Omega, \Lambda \rangle \not\supseteq \langle \Omega', \Lambda' \rangle$.

This also means that the empty network of alignments $\langle \emptyset, \emptyset \rangle$ (containing no ontology and no alignment) is subsumed by any other network of ontologies and that it is always possible to reduce a network of ontologies to one made of a single ontology (gathering all axioms of the ontologies) and a single alignment (containing all correspondences) $\langle \{\dot{o} = \cup_{o \in \Omega} o\}, \{A_{\dot{o}, \dot{o}} = \cup_{A \in \Lambda} A\} \rangle$.

From subsumption, conjunction (meet) can be introduced in a standard way:

Definition 55 (Syntactic conjunction of networks of ontologies). *Given a finite family of networks of ontologies, $\{\langle \Omega_i, \Lambda_i \rangle\}_{i \in I}$, $\prod_{i \in I} \langle \Omega_i, \Lambda_i \rangle = \langle \Omega', \Lambda' \rangle$ such that $\forall i \in I, \langle \Omega', \Lambda' \rangle \sqsubseteq \langle \Omega_i, \Lambda_i \rangle$ and $\forall \langle \Omega'', \Lambda'' \rangle; \langle \Omega'', \Lambda'' \rangle \sqsubseteq \langle \Omega_i, \Lambda_i \rangle, \langle \Omega'', \Lambda'' \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$.*

Such a conjunction always exists because the empty network of ontologies is subsumed by all network of ontologies. It remains to prove that it is unique (and it is certainly not unique because we are dealing with syntax, aren't we?).

The semantic web itself can be seen as a network of ontologies. Our goal is to make sense of these, i.e., to give them a semantics.

Chapter 10

Alignment algebra

The next important component of the alignment is the relation that holds between the entities. We identify a set of relations Θ that is used for expressing the relations between the entities. Matching algorithms primarily use the equivalence relation ($=$) meaning that the matched objects are the same or are equivalent if these are formulas. It is possible to use relations from the ontology language within Θ . For instance, using OWL, it is possible to take advantage of the `owl:equivalentClass`, `owl:disjointWith` or `rdfs:subClassOf` relations in order to relate classes of two ontologies. These relations correspond to set-theoretic relations between classes: *equivalence* ($=$), *disjointness* (\perp), *less general* (\sqsubseteq). They can be used without reference to any ontology language.

Example 30 (Alignment). *Consider two alignments A_1 and A_2 , relating respectively the German to the French ontology and the French ontology to the British one, containing the following correspondences (A_1 is on the left, A_2 on the right):*

$$\begin{array}{ll} \text{Konstruktion} \perp \text{Commune} & \text{Commune} \geq \text{Municipality} \\ \text{Stadtgebiet} > \text{Ville} & \text{Ville} \not\subseteq \text{Municipality} \end{array}$$

*This means that A_1 considers that a *Konstruktion*, i.e., a *Building*, is disjoint from a *Commune*, i.e., a *Ward*, and a *Stadtgebiet*, i.e., a *Urban area*, is more general than a *Ville*, i.e., a *Town*. A_2 expresses that a *Commune* is more general or equivalent to a *Municipality* and *Ville* overlaps with *Municipality*, i.e., that both concepts have common instances but none is more general than the other.*

This definition does not tell how to interpret this set of correspondences. However, it is clear from usage that it has to be interpreted in a conjunctive manner: all the correspondences are asserted to hold when asserting an alignment.

Hence, the problem of expressing disjunctions of correspondences can be raised. This can be because it is necessary to aggregate the result of methods which address the ontology matching problem from different dimensions, this can be because the person or the program generating the alignment is unsure about the exact relation but knows that this relation is constrained to a specific set of alternative relations.

Example 31 (Disjunctive relations). *For instance, an engineer may know that a *Stadt*, i.e., *Town*, and a *Town* are similar things but may not know exactly the nature of the overlaps. She can express that they are not disjoint by the disjunction of relations $<$, $>$, $\not\subseteq$ and $=$, thus*

prohibiting \perp . This can also be because the alignment has been generated by composing two alignments. This operation does not usually return a simple relation but a disjunction of such relations, e.g., if *Stadtgebiet*, i.e., Urban area, is more general than *Ville*, i.e., Town, and *Ville* overlaps *Municipality*, then *Stadtgebiet* either is more general or overlaps *Municipality*, it cannot be disjoint with it. Hence, the result is a disjunction of relations.

This is also the case of the \geq (more-general-or-equal) and \leq (more-specific-or-equal) relations used by some systems. These are typically the disjunction of $<$ and $=$ or $>$ and $=$. In fact, practice which considers that if both \leq and \geq hold (conjunction), then $=$ holds, only reflects the set operation: $\{<, =\} \cap \{>, =\} = \{=\}$ or the logical interpretation that:

$$\forall a, b, (a < b \vee a = b) \wedge (a > b \vee a = b) \models a = b \text{ if } <, > \text{ and } = \text{ are exclusive}$$

In order to apply a systematic treatment for disjunctive alignment relations, we use algebra of relations and we show that this has many advantages.

10.1 Relation algebra

An algebra of binary relations (hereafter referred to as relation algebra¹) [Tarski, 1941] is a structure $\langle \Theta, \wedge, \vee, \neg, \top, \perp, *, \bar{\cdot}, 1' \rangle$ such that $\langle \Theta, \wedge, \vee, \neg, \top, \perp \rangle$ is a Boolean algebra; $*$ is an associative internal composition law with (left and right) unity element $1'$, that distributes over \vee ; $\bar{\cdot}$ is an internal involutive unary operator, that distributes over \vee, \wedge and $*$.

We consider a particular type of relation algebras² in which Θ is the powerset of a generating set Γ closed under \neg and $\wedge/\vee/\neg$ are set intersection/union/complementation ($\cap/\cup/\Gamma-$). Such an algebra of (binary) relations is defined by $\langle 2^\Gamma, \cap, \cup, \Gamma-, \Gamma, \emptyset, \cdot, {}^{-1}, \{=\} \rangle$ such that:

- Γ is a set of jointly exhaustive and pairwise disjoint (JEPD) relations between two entities. This means that, in any situation, the actual relation between two objects is one and only one of these relations. Sets of relations allow to express uncertainty: the full Γ set is the "I do not know" relation since it is satisfied by any pair of entities;
- \cap and \cup are set operations used to meet and join two sets of base relations, hence if xry or $xr'y$, then $xr \cup r'y$;
- \cdot is the composition operator such that if xry and $y r' z$, then $x r \cdot r' z$; " $=$ " is such that $\forall r \in \Gamma, (r \cdot =) = (= \cdot r) = r$;
- ${}^{-1}$ is the converse operator, i.e., such that $\forall e, e' \in \Gamma, ere' \Leftrightarrow e'r^{-1}e$.

These operations are applied to sets of base relations by distributing them on each element, e.g., $R \cdot R' = \bigcup_{r \in R, r' \in R'} r \cdot r'$.

A typical example of such an algebra is the Allen algebra of temporal interval relations [Allen, 1983]. Here, we will consider, as an example, a simpler algebra, called *A5*, isomorphic to that applying to sets in which the base (JEPD) relations between two sets are equivalent ($=$), includes ($>$), is-included-in ($<$), overlaps (\overlapping) and disjoint (\perp). In this algebra, all base relations but $<$ and $>$ are their own converse while $>^{-1} = <$ and $<^{-1} = >$.

¹There was a mistake in the presentation of [Euzenat, 2008] which merged negation and inverse.

²[Ligozat and Renz, 2004] shows that a weaker structure than algebras of relations, non associative, can be used in most of the purposes of qualitative calculi. However, we will need associativity later.

The complete set of $2^5 - 1 = 31$ valid relations that can be made out of these 5 base relations is depicted in Figure 10.1. Among these relations, Γ means "I do not know" as it contains all the base relations. \neq is equivalent to $\{\langle, \rangle, \bar{\neq}, \perp\}$, \leq is equivalent to $\{=, \langle\}$, \geq to $\{=, \rangle\}$, \approx to $\{\langle, \rangle, \bar{\neq}\}$ and $\not\leq$ to $\{=, \langle, \rangle, \bar{\neq}\}$. The composition table is given in Table 10.1.

Relation algebras can still be used when the ontology entities are formulas (or queries) and the base relations are logical connectives between formulas (\Rightarrow, \equiv). Indeed, it is sufficient to split the disjunctive relations into a disjunction of formulas:

$$\phi\{\Rightarrow, \equiv\}\psi \text{ would be equivalent to } \phi \equiv \psi \vee \phi \Rightarrow \psi$$

(these relations are not exclusive anymore).

10.2 Aggregating matcher results

The set Θ is closed for \vee, \wedge and \neg . This means that any combination of these operations yields an element of Θ . This is very powerful if one wants to combine relations, e.g., for combining correspondences and alignments.

When matching methods bring new evidences for a correspondence from a different perspective, they are thought of as bringing new arguments in favor of a correspondence. Hence, its results must be aggregated with union ($R \cup R'$). When the matching methods, instead, are competing algorithms providing all the possible base relations, i.e., providing arguments against the non selected correspondences, then intersection ($R \cap R'$) should be used. Because, we now have two distinguished operations, they can be used together in the same application.

These operations can be used for describing two cases of matching process: an expanding matching process which starts with the empty relation (\emptyset) between each pair of entities and which finds evidences for more base relations between these entities aggregating them with \cup and a contracting process which starts with Γ between each pair of entities and which discards support for some base relations between entities aggregating the result with \cap . In the first case, the more matching methods are used, the less precise the alignment becomes: this can be balanced by confidence measures as we will see below. In the second case, the more methods are used, the more precise the alignment becomes.

These operations on Θ are used in correspondence aggregation: an alignment is interpreted as all the correspondences it contains hold, and a distributed system is interpreted as all alignments hold. Hence, the disjunctive aggregation of alignments is based on the combination of their set of correspondences with the union of relations; the conjunctive aggregation of alignments is based on the combination of their set of correspondences with the intersection of relations. Hence, we define a normalisation operation \bar{A} , which implements the conjunctive interpretation of alignments. It provides exactly one correspondence per pair of entities and makes explicit all the relations between entities in A :

$$\begin{aligned} A^0 &= \{\langle e, e', \Gamma \rangle \mid e \in Q_L(o), e' \in Q_{L'}(o')\} \\ \bar{A} &= \{\langle e, e', \cap_{\langle e, e', r \rangle \in AUA^0 r} \rangle\} \end{aligned}$$

It is then easy to define intersection:

$$A \wedge A' = \{\langle e, e', r \cap r' \rangle \mid \langle e, e', r \rangle \in \bar{A}, \langle e, e', r' \rangle \in \bar{A}'\}$$

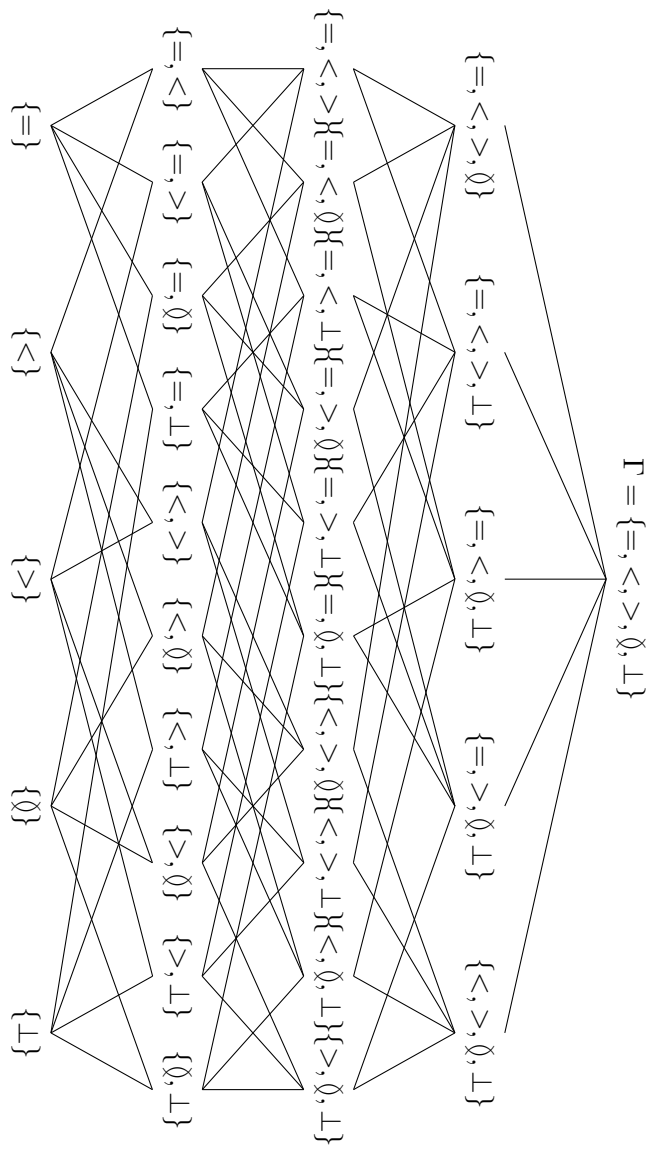


Figure 10.1: The lattice of 31 disjunctive relations in A_5 .

as well as additional operators such as disjunction and converse of alignments:

$$A \vee A' = \{\langle e, e', r \cup r' \rangle \mid \langle e, e', r \rangle \in \bar{A}, \langle e, e', r' \rangle \in \bar{A}'\}$$

$$A^{-1} = \{\langle e', e, r^{-1} \rangle \mid \langle e, e', r \rangle \in \bar{A}\}$$

Below, alignments are always presented in a reduced normalised way, i.e., without trivial $\langle e, e', \Gamma \rangle$ correspondences added by normalisation.

Example 32 (Alignment aggregation). *Consider two alignments A_3 and A_5 , resulting from two different matchers which match ontologies using different features for ruling out correspondences (A_3 is on the left, A_5 on the right):*

$$\begin{array}{ll} \text{Konstruktion}\{\perp\}\text{Municipality} & \text{Stadt}\{\langle\}\text{Town} \\ \text{Stadtgebiet}\{\>, \emptyset\}\text{Municipality} & \text{Stadtgebiet}\{\perp, \emptyset\}\text{Municipality} \end{array}$$

Since these matchers provide competing alignments between ontology o and o' , their result can be aggregated conjunctively. The result $A_6 = A_3 \wedge A_5$, is given below as the left-hand side alignment:

$$\begin{array}{ll} \text{Konstruktion}\{\perp\}\text{Municipality} & \text{Stadt}\{\langle\}\text{Town} \\ \text{Stadtgebiet}\{\emptyset\}\text{Municipality} & \text{Stadt}\{=, \langle, \>, \emptyset\}\text{Town} \\ & \text{Stadtgebiet}\{\perp\}\text{Municipality} \end{array}$$

This alignment is aggregated with the right-hand side alignment A_4 . Since they provide evidence for alignments from different perspective, they are aggregated disjunctively, yielding $A_8 = A_4 \vee A_6$:

$$\begin{array}{l} \text{Stadt}\{=, \langle, \>, \emptyset\}\text{Town} \\ \text{Stadtgebiet}\{\emptyset, \perp\}\text{Municipality} \end{array}$$

Algebras of relations are useful because they can account for these two behaviours. However, there are other benefits brought by algebra of relations.

10.3 Composing alignments

Another way of reusing alignments is to deduce new alignments from existing ones. One way to do so, is to compose alignments. If there exists an alignment between ontology o and ontology o'' , and another alignment between o'' and a third ontology o' , we would like to find which correspondences hold between o and o' . The operation that returns this set of correspondences is called composition.

Alignment composition has already considered [Zimmermann *et al.*, 2006] with the idea that, in an open system like the Alignment API, the rules for composing alignment relations, e.g., $\text{instanceOf} \cdot \text{subClassOf} = \text{instanceOf}$, should be given by a composition table. Composition tables come directly from algebra of relations and they naturally extend from base relations to disjunctions of base relations.

Alignment composition can thus be reduced to combining correspondences with regard to their relations and the structure of related entities and computing the confidence degree of the result. The composition table between the base relations of A_5 is given in Table 10.1.

	=	<	<	∅	⊥
=	=	>	<	∅	⊥
>	>	>	><=∅	>∅	>∅ ⊥
<	<	Γ	<	<∅ ⊥	⊥
∅	∅	>∅ ⊥	<∅	Γ	>∅ ⊥
⊥	⊥	⊥	<∅ ⊥	<∅ ⊥	Γ

 Table 10.1: Composition table for the $A5$ relation algebra.

The composition of two alignments A and A' is defined by:

$$A \cdot A' = \overline{\{\langle e, e'', r \cdot r' \rangle \mid \langle e, e', r \rangle \in A, \langle e', e'', r' \rangle \in A'\}}$$

One can compose an alignment with itself (self-composition) through: $A^2 = A \cdot A^{-1} \cdot A$. This operation may provide new correspondences.

Example 33 (Composing alignments). *The alignment A_3 of Example 32, is the result of the composition of alignments A_1 and A_2 of Example 30: $A_3 = A_1 \cdot A_2$. The first simple application of Table 10.1 occurs when composing *Konstruktion* $\{\perp\}$ *Commune* and *Commune* $\{>, =\}$ *Municipality*, then it can be deduced that *Konstruktion* $\{\perp\}$ *Municipality* because $\{\perp\} \cdot \{>, =\} = (\perp \cdot >) \cup (\perp \cdot =) = \{\perp\}$. Things can be more complex, when composing *Stadtgebiet* $\{>\}$ *Ville* and *Ville* $\{\emptyset\}$ *Municipality*, then Table 10.1 allows to deduce that *Stadtgebiet* $\{>, \emptyset\}$ *Municipality* because $\{>\} \cdot \{\emptyset\} = \{>\} \cdot \emptyset = \{<, \emptyset\}$. The result provided by the table in this case is a disjunction of relations because it is not possible to obtain more precise information from the alignments alone.*

Very often the composition of two base relations is not a base relation but a disjunction of relations. Hence, if we were not dealing with sets of base relations, it would not be possible to represent the composition of two alignments by an alignment.

Moreover, defining composition by an algebra of relations automatically satisfies all the constraints on the categorical characterisation of alignments defined in [Zimmermann *et al.*, 2006]: it must be associative and have an identity element. This is true from the definition of algebra of relations.

10.4 Algebraic reasoning with alignments

α -consequences are correspondences which are entailed by two aligned ontologies [Euzenat, 2007]; they can be extended as the correspondences entailed by a system of many ontologies and many alignments between them. [Zimmermann and Le Duc, 2008] introduced the notion of quasi-consequences as the set of formulas entailed by a set of alignments alone (without considering ontologies). This notion can be straightforwardly extended to correspondences as quasi- α -consequences: the correspondences which are entailed by the set of alignments, when considering the ontologies as void of axioms. Quasi- α -consequences are also α -consequences.

Reasoning on alignments aims at using existing alignments in order to deduce more and more complete alignments. Such a reasoning procedure can be considered, for soundness and completeness, with respect to α -consequences.

Algebraic reasoning (using combination of composition, converse, and intersection) can be used as a practical and efficient way to reason with alignments. The algebraic closure of a set of alignments S is the set of normalised alignments, containing S , closed under composition, converse and intersection.

This procedure is correct (the algebraic operations can be transformed into their logical equivalent). However, since it does not consider ontologies, it can only deduce quasi- α -consequences. We have no guarantee that it is complete even for finding quasi- α -consequences.

However, this can already be used for two purposes: (1) improving the existing alignments by deducing new correspondences coming either from the alignment itself or from other alignments, and (2) checking the consistency of a set of alignments (or one alignment). Indeed, if we can deduce $x\{y\}$, e.g., because $x\{<\}y$ and $x\{>\}y$ for two competing matchers, since the intersection is empty we know that the alignment itself is inconsistent. This kind of reasoning can be more complex, involving several alignments as well as composition operations, i.e., checking a whole distributed system. Then, the set of alignments as a whole would be inconsistent, hence the distributed system would have no model.

Example 34 (Algebraic reasoning). *The simplest instance of an inconsistent alignment is to have two contradictory statements like $\text{Konstruktion}\{\perp\}\text{Town}$ and $\text{Konstruktion}\{<\}\text{Town}$ in the same alignment. The conjunction of these two relations, obtained by normalisation, is empty. Such an inconsistent alignment can also be obtained by combining consistent alignments. For instance, aggregating conjunctively alignments A_3 and A_4 of Example 32, will generate the inconsistent $\text{Stadtgebiet}\{\}\text{Municipality}$ correspondence.*

Algebraic reasoning allows to expand alignments. For instance, $\text{Stadt}\{>\}\text{Town}$, $\text{Stadtgebiet}\{<\}\text{Town}$, $\text{Stadtgebiet}\{\perp\}\text{Municipality}$ entails $\text{Stadt}\{>, \emptyset, \perp\}\text{Municipality}$. Computing the compositional closure of this alignment will find this correspondence. Moreover, if the initial alignment also contain $\text{Stadt}\{=, <\}\text{Municipality}$, the compositional closure will bring the inconsistency to light.

Once again, it is possible to use disjunctive relations to better evaluate alignments. Indeed, the problem is that if a matcher returns a correspondence between two entities with relation \leq while the expected (and exact) relation was $<$, then the use of syntactic precision and recall measures would count this relation as incorrect. Hence, if the expected alignment was made of this correspondence alone, both precision and recall would be 0. This is unfair because it cannot be said that this correspondence is both incorrect and incomplete. In fact, it is incomplete, because it does not provide the exact relation, but not incorrect, because the relation is more general than the correct one.

This is indeed what happens with semantic precision and recall [Euzenat, 2007]: since the relation $\{<, =\}$, which is the disjunction of $<$ and $=$, can be deduced from $\{<\}$ alone, it would count as correct for semantic precision and still as incorrect for semantic recall because $\{<, =\}$ does not entail $\{<\}$.

We could introduce an algebraic precision and recall for evaluating ontology alignments as an intermediary step between classic precision and recall and semantic precision and recall. It would simply use the inclusion between the relations as suggested above instead of the entailment between correspondences of [Euzenat, 2007] and would be far easier to compute. The resulting measure would be a relaxation of precision and recall in the sense of [Ehrig and Euzenat, 2005].

10.5 Algebra granularity

In order to investigate granularity within algebras of relations, we introduced the notion of weakening [Euzenat, 2001]. Weakening an algebra of relations simply consists of grouping together several base relations and taking the result as the base relations of the, less precise, weaker algebra.

In fact, taking any maximal antichain³ that preserves converse in the lattice of Figure 10.1 yields a base for an algebra of relations. Other constraints can be put on weakening, such as requiring that they preserve a neighbourhood structure. Neighbourhood structures for algebras of relations have been introduced in [Freksa, 1992]. They are based on a connectivity relation between relations that is used for defining neighbourhood. This connectivity relation can be based on different properties of the domain the relations apply to. We have shown that granularity operators, at least in time and space algebras, can be automatically built on such neighbourhoods [Euzenat, 2001].

In terms of alignment, the interesting aspect of this weakening operation is that it helps considering that alignments using different sets of relations are compatible and can still be used together. Coming back to the example of set-relations, there can be systems that provides only the = base relation leaving implicitly all the others as Γ , there are other systems like the one considered previously which consider =, <, >, \emptyset and \perp . The set of base relations is different and thus it is not easy to combine two such alignments. However, if we consider that the first one is a weakening of the second one (grouping <, >, \emptyset and \perp into \neq), then it is possible to import one alignment into another formalism and vice-versa (at the expense of completeness when we export to the weaker algebra).

Example 35 (Algebra granularity). *Different sources of alignments may provide alignments with different kinds of relations between objects. For instance, the following A_7 alignment (left-hand side) is expressed in the simple $\{\perp, \neq\}$ algebra, i.e., identifying only incompatible elements.*

$$\begin{array}{cc} \text{Stadt}\{\neq\} \text{Town} & \text{Stadt}\{\neq\} \text{Town} \\ \text{Stadtgebiet}\{\perp\} \text{Municipality} & \text{Stadtgebiet}\{\perp, \neq\} \text{Municipality} \end{array}$$

Thanks to the use of compatible algebras, A_7 can be expressed in the more expressive algebra. In fact, the alignment A_4 of Example 32 is the transcription of A_7 in the A_5 algebra. On the other hand, it is possible to degrade an alignment into the coarser algebra at the expense of precision. The alignment on the right-hand side is the result of converting the alignment A_5 of Example 32 to the $\{\perp, \neq\}$ algebra.

Figure 10.2 shows the “interesting” weakened algebras of relations from the initial algebra. It features the $\{=, \neq\}$ algebra but also shows that the usually considered $\{=, \leq, \geq, \perp\}$ is not a correct base for such an algebra because it is neither jointly exhaustive (\leq and $=$ can occur at the same time), nor pairwise disjoint (\emptyset is missing).

³An antichain is a set of relations such that no one is comparable to the other.

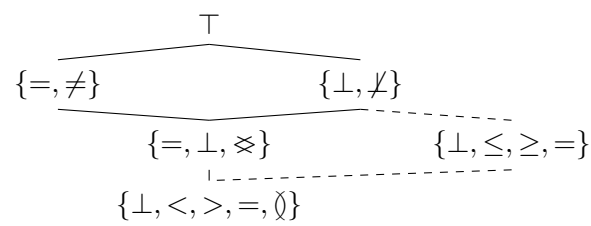


Figure 10.2: The reasonable weakenings of $A5$ ($\neq = \{<, >, =, \emptyset\}$ and $\infty = \{<, >, \emptyset\}$).

Chapter 11

Alignment semantics

If alignments have to be used with ontologies, they have to be given a semantics. The problem is to provide a semantics to aligned ontologies or networks of ontologies. This semantics must play correctly with that of ontologies, because connecting ontologies to other ontologies should be compatible with the semantics of ontologies.

Intuitively, an alignment expresses constraints between ontologies. It thus restricts the semantics of these ontologies to satisfy these constraints. This is exactly the general semantic framework that we consider here. The alignment will restrict the ontology models which are compatible with the alignment.

When ontologies are independent, i.e., not related with alignments, it is natural that their semantics be the classical semantics for these ontologies, e.g., a set of models $\mathcal{M}(o)$. A model is a function m from the elements of the ontologies to a particular domain Δ . Very often, this domain Δ contains the powerset and the product of a particular universe of discourse (classes are interpreted on the powerset and relations on the product).

Because relations in Θ are not part of any ontologies, their semantics is given independently from the semantics of ontologies. Here we postulate that there exist a global domain of interpretation U which contains the union of the domains of interpretation of each individual ontology. This point is certainly the main issue in the semantics of networked ontologies, it is often treated in different ways depending on the logical options: considering that all ontologies have the same domain of interpretation, considering that they have disjoint domains of interpretation, or not imposing constraints. Here we do not impose constraints.

Definition 56 (Interpretation of alignment relations). *Given an alignment relation $r \in \Theta$ and a global domain of interpretation U , r is interpreted as a binary relation over U , i.e., $r^U \subseteq U \times U$.*

Typically, between classes interpretations, $=$ is interpreted as set equality, \leq as set inclusion, etc.

Once these ontologies are brought together through an alignment, the correspondences in the alignment impose constraints to the set of acceptable models, i.e., it selects those models which are compatible with the models of the other ontologies through the alignment.

Hence, the semantics of two aligned ontologies may be given as a set of models which are pairs of compatible models. The semantics of ontologies would then be given with regard to the semantics of relations as:

Definition 57 (Satisfied correspondence). *A correspondence $\mu = \langle e, e', r \rangle$ is satisfied by two*

models m, m' of o, o' if and only if

$$\langle m(e), m'(e') \rangle \in r^U$$

This is denoted as $m, m' \models \mu$.

Example 36 (Interpretation of relations). Typically, in the language used as example, if m and m' are respective models of o and o' :

$$\begin{aligned} m, m' \models \langle c, c', = \rangle &\text{ iff } m(c) = m'(c') \\ m, m' \models \langle c, c', \subseteq \rangle &\text{ iff } m(c) \subseteq m'(c') \\ m, m' \models \langle c, c', \supseteq \rangle &\text{ iff } m(c) \supseteq m'(c') \\ m, m' \models \langle c, c', \perp \rangle &\text{ iff } m(c) \cap m'(c') = \emptyset \end{aligned}$$

Definition 58 (Models of alignments). Given two ontologies o and o' and an alignment A between these ontologies, a model of this alignment is a pair $\langle m, m' \rangle \in \mathcal{M}(o) \times \mathcal{M}(o')$, such that $\forall \mu \in A, m, m' \models \mu$ (denoted $m, m' \models A$).

Example 37 (Model of an alignment). A model for the alignment $A_{1,2}$ of Figure 9.2, is any pair of models $\langle m_1, m_2 \rangle$ of ontology o_1 and o_2 such that $m_1(b_1) \subseteq m_2(d_2)$.

Models of networks of ontologies are an extension of the models of alignments. They select vectors of compatible models of each ontology in the network. Compatibility consists of satisfying all the alignments of the network.

Definition 59 (Models of networks of ontologies). Given a networks of ontologies $\langle \Omega, \Lambda \rangle$ with $n = |\Omega|$, a model of $\langle \Omega, \Lambda \rangle$ is a n -uple of models $\langle m_1 \dots m_n \rangle \in \mathcal{M}(o_1) \times \dots \mathcal{M}(o_n)$, such that for each alignment $A \in \Lambda(o_i, o_j), m_i, m_j \models A$.

In that respect, alignments act as model filters for the ontologies. They select the ontology interpretations which are coherent with the alignments. This allows for transferring information from one ontology to another since reducing the set of models will entail more consequences in each aligned ontology.

Example 38 (Model of a network of ontologies). Hence, a model for the network of ontologies of Figure 9.2, is any triple of models $\langle m_1, m_2, m_3 \rangle$ of ontology o_1, o_2 and o_3 such that $m_3(e_3) \subseteq m_1(b_1) \subseteq m_2(d_2), m_2(c_2) \subseteq m_3(b_3)$ and $m_3(f_3) \subseteq m_1(e_1)$.

Such models remain very distributed. Indeed, they tell which models of each ontology are selected and, by extension, what are the consequences of an ontology. However, since there is no explicit link between the local models: the constraints that have been imposed by the alignments are not recorded in the model structure.

Different semantics provide alternative ways to record the constraints imposed by alignments: through relations between domains of interpretation [Ghidini and Serafini, 1998; Borgida and Serafini, 2003], through equalising functions (see §12 [Zimmermann and Euzenat, 2006; Zimmermann, 2008]), by imposing equal [Lenzerini, 2002] or disjoint [Grau *et al.*, 2006] domains. Here, it is not useful to record such constraints, so we will remain independent from such particular constraints and will only rely on tuples of compatible models. So, the definition of consequence that we use can be computed with one of the alternative definitions.

Property 16 (Monotony of consequence).

$$\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle \Rightarrow \mathcal{M}(\langle \Omega', \Lambda' \rangle) \subseteq \mathcal{M}(\langle \Omega, \Lambda \rangle)$$

Proof. $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$ implies that $\exists \langle h, k \rangle; \forall o \in \Omega, o \subseteq h(o)$ and $\forall A \in \Lambda(o, o'), A \subseteq k(A) \wedge k(A) \in \Lambda'(h(o), h(o'))$. Hence, $\forall m \in \mathcal{M}(h(o)), m \in \mathcal{M}(o)$ and $\forall \langle m, m' \rangle \in \mathcal{M}(h(o)) \times \mathcal{M}(h(o')), m, m' \models k(A) \Rightarrow m, m' \models A$ (because m and m' are models of o and o' and A contains less constraints to satisfy than $k(A)$). Thus, any model of $\langle \Omega', \Lambda' \rangle$ is also a model of $\langle \Omega, \Lambda \rangle$. In addition, $\langle \Omega', \Lambda' \rangle$ may contain more alignments (and eventually more ontologies) which may further reduce its set of models. \square

11.1 Consistence, consequence and closure

A network of ontologies is consistent if it has a model. By extension, an ontology or an alignment will be consistent within a network of ontologies if the network of ontologies is consistent. This implies that an ontology, consistent when taken in isolation, can be inconsistent in a network of ontologies. More disturbing, if one of the ontologies in the network is inconsistent, then the network as a whole is inconsistent.

There is no consequence relation for a network of ontologies. In fact, we have not defined what it means for a formula to be the consequence of a network. We define two notions of consequences called ω -consequence and α -consequence.

α -consequences are those consequences of aligned ontologies which are correspondences.

Definition 60 (α -Consequence of networks of ontologies). *Given a finite set of n ontologies Ω and a finite set of alignments Λ between pairs of ontologies in Ω , a correspondence μ between two ontologies o_i and o_j in Ω is an α -consequence of $\langle \Omega, \Lambda \rangle$ (noted $\langle \Omega, \Lambda \rangle \models \mu$) if and only if for all models $\langle m_1, \dots, m_n \rangle$ of $\langle \Omega, \Lambda \rangle$, $m_i, m_j \models \mu$ (the set of α -consequences between o_i and o_j is denoted by $Cn_{\Omega, \Lambda}^{\alpha}(o_i, o_j)$).*

Example 39 (α -consequences). *The closure of $A_{1,3}$ is:*

$$Cn_{\Omega, \Lambda}^{\alpha}(o_1, o_3) = \left\{ \begin{array}{l} e_1 \geq f_3 \quad b_1 \geq e_3 \\ c_1 \geq f_3 \quad a_1 \geq f_3 \\ a_1 \leq e_3 \quad b_1 \leq b_3 \\ b_1 \leq a_3 \quad b_1 \perp c_3 \\ b_1 \perp d_3 \quad b_1 \perp e_3 \end{array} \right.$$

while if the network is reduced to the two involved ontologies only, the closure would only be:

$$Cn_{\{o_1, o_3\}, \{A_{1,3}\}}^{\alpha}(o_1, o_3) = \left\{ \begin{array}{l} e_1 \geq f_3 \quad b_1 \geq e_3 \\ c_1 \geq f_3 \quad a_1 \geq f_3 \\ a_1 \leq e_3 \end{array} \right.$$

It is thus clear that the connecting to more ontologies provide more information.

α -consequences of an alignment are defined as the α -consequences of the network made of this alignment and the two ontologies it connects (denoted $A \models \mu$). The α -consequences of a particular alignment are usually larger than the alignment ($A \subseteq Cn^{\alpha}(A)$). If the alignment is not satisfiable, then any correspondence is one of its α -consequences. The α -closure is the

set of α -consequences: the correspondences which are satisfied in all models of the network of ontologies.

Similarly, the ω -consequences of an ontology in a network are formulas that are satisfied in all models of the ontology in the network.

Definition 61 (ω -Consequence of an ontology in a networks of ontologies). *Given a finite set of n ontologies Ω and a finite set of alignments Λ between pairs of ontologies in Ω , a formula δ in the ontology language $o_i \in \Omega$ is an ω -consequence of o_i in $\langle \Omega, \Lambda \rangle$ (noted $\langle \Omega, \Lambda \rangle, o_i \models \delta$) if and only if for all models $\langle m_1, \dots, m_n \rangle$ of $\langle \Omega, \Lambda \rangle$, $m_i \models \delta$ (the set of ω -consequences of o_i is denoted by $Cn_{\Omega, \Lambda}^{\omega}(o_i)$).*

The ω -closure of an ontology is the set of its ω -consequences. We may also use the notation $Cn_{\langle \Omega, \Lambda \rangle}^{\alpha}$ for $Cn_{\Omega, \Lambda}^{\alpha}$ and $Cn_{\langle \Omega, \Lambda \rangle}^{\omega}$ for $Cn_{\Omega, \Lambda}^{\omega}$.

These ω -consequences are larger than the classical consequences of the ontology ($Cn(o) \subseteq Cn_{\Omega, \Lambda}^{\omega}(o)$). This definition is semantic: usually, closure operators are defined syntactically.

Example 40 (ω -consequences). *The simple consequences of the ontology o_3 are:*

$$Cn(o_3) = \left\{ \begin{array}{ll} b_3 \sqsubseteq a_3 & c_3 \sqsubseteq a_3 & g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3 & e_3 \sqsubseteq c_3 & f_3 \sqsubseteq b_3 \\ i \in e_3 & b_3 \perp c_3 & f_3 \sqsubseteq a_3 & g_3 \sqsubseteq a_3 \\ d_3 \sqsubseteq a_3 & e_3 \sqsubseteq a_3 & i \in c_3 \\ i \in a_3 & d_3 \perp b_3 & e_3 \perp b_3 \\ f_3 \perp c_3 & g_3 \perp c_3 \end{array} \right.$$

while within the networks of ontologies, there are even more consequences:

$$Cn_{\Omega, \Lambda}^{\omega}(o_3) = \left\{ \begin{array}{ll} b_3 \sqsubseteq a_3 & c_3 \sqsubseteq a_3 & g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3 & e_3 \sqsubseteq c_3 & f_3 \sqsubseteq b_3 \\ i \in e_3 & b_3 \perp c_3 & f_3 \sqsubseteq a_3 & g_3 \sqsubseteq a_3 \\ d_3 \sqsubseteq a_3 & e_3 \sqsubseteq a_3 & i \in c_3 \\ i \in a_3 & d_3 \perp b_3 & e_3 \perp b_3 \\ f_3 \perp c_3 & g_3 \perp c_3 & b_3 \supseteq e_3 \\ i \in b_3 \end{array} \right.$$

We can now define the closure of a network of ontologies by the network of ontologies which replaces each ontology by its ω -closure and each alignment by its α -closure:

$$Cn(\langle \Omega, \Lambda \rangle) = \langle \{Cn_{\Omega, \Lambda}^{\omega}(o)\}_{o \in \Omega}, \{Cn_{\Omega, \Lambda}^{\alpha}(o, o')\}_{o, o' \in \Omega} \rangle$$

Property 17. *Cn is a closure operation¹.*

Proof. $\forall \langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$ normalised networks of ontologies:

- $\langle \Omega, \Lambda \rangle \sqsubseteq Cn(\langle \Omega, \Lambda \rangle)$, because $\forall \langle h, k \rangle; \forall o_i, o_j \in \Omega, h(o_i) = Cn_{\Omega, \Lambda}^{\omega}(o_i)$ and $k(\lambda(o_i, o_j)) = Cn_{\Omega, \Lambda}^{\alpha}(o_i, o_j)$, $o_i \subseteq Cn_{\Omega, \Lambda}^{\omega}(o_i)$, because $\forall \delta \in o, \forall m \in \mathcal{M}(\langle \Omega, \Lambda \rangle), m_i \models o_i$, hence $m_i \models \delta$, so $\delta \in Cn_{\Omega, \Lambda}^{\omega}(o_i)$; and $\lambda(o_i, o_j) \subseteq Cn_{\Omega, \Lambda}^{\alpha}(o_i, o_j)$, because $\forall \mu \in \lambda(o_i, o_j), \forall m \in \mathcal{M}(\langle \Omega, \Lambda \rangle), m_i, m_j \models \mu$, hence $\mu \in Cn_{\Omega, \Lambda}^{\alpha}(\lambda(o_i, o_j))$.

¹A closure operation in a set S satisfies three properties: $\forall X, Y \in S : X \subseteq Cn(X), Cn(X) = Cn(Cn(X))$, and $X \subseteq Y \Rightarrow Cn(X) \subseteq Cn(Y)$.

- $\forall m \in \mathcal{M}(\langle \Omega, \Lambda \rangle)$, (i) $\forall o_i \in \Omega, m_i \in \mathcal{M}(o_i)$ and $\forall o_i, o_j \in \Omega, m_i, m_j \models \lambda(o_i, o_j)$, (ii) $\forall o_i \in \Omega, m_i \in \mathcal{M}(Cn_{\Omega, \Lambda}^{\omega}(o_i))$, and (iii) $\forall o_i, o_j \in \Omega, m_i, m_j \models Cn_{\Omega, \Lambda}^{\alpha}(o_i, o_j)$ (the two last assertions because the closure contains elements true in all models). Hence, $m \in \mathcal{M}(Cn(\langle \Omega, \Lambda \rangle))$. So, $\mathcal{M}(\langle \Omega, \Lambda \rangle) \subseteq \mathcal{M}(Cn(\langle \Omega, \Lambda \rangle))$, and thus (less models means more consequences) $Cn(\langle \Omega, \Lambda \rangle) \supseteq Cn(Cn(\langle \Omega, \Lambda \rangle))$. By the first clause, we know that $Cn(\langle \Omega, \Lambda \rangle) \sqsubseteq Cn(Cn(\langle \Omega, \Lambda \rangle))$, so, $Cn(\langle \Omega, \Lambda \rangle) \equiv Cn(Cn(\langle \Omega, \Lambda \rangle))$.
- If $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$, then $\mathcal{M}(\langle \Omega', \Lambda' \rangle) \subseteq \mathcal{M}(\langle \Omega, \Lambda \rangle)$ by Property 16. Thus, $\forall o \in \Omega, Cn_{\Omega, \Lambda}^{\omega}(o) \subseteq Cn_{\Omega', \Lambda'}^{\omega}(h(o))$ and $\forall A \in \Lambda(o, o'), Cn_{\Omega, \Lambda}^{\alpha}(A) \subseteq Cn_{\Omega', \Lambda'}^{\alpha}(k(A))$. Hence, $Cn(\langle \Omega, \Lambda \rangle) \sqsubseteq Cn(\langle \Omega', \Lambda' \rangle)$.

□

Example 41 (Full network closure). *Here is the closure of the network of ontologies of Example 29:*

$$\begin{aligned}
 Cn_{\Omega, \Lambda}^{\omega}(o_1) = Cn(o_1) &= \left\{ \begin{array}{ll} b_1 \sqsubseteq a_1 & c_1 \sqsubseteq a_1 \\ d_1 \sqsubseteq c_1 & e_1 \sqsubseteq c_1 \\ d_1 \sqsubseteq a_1 & e_1 \sqsubseteq a_1 \end{array} \right. \\
 Cn_{\Omega, \Lambda}^{\omega}(o_2) = Cn(o_2) &= \left\{ \begin{array}{lll} b_2 \sqsubseteq a_2 & c_2 \sqsubseteq a_2 & g_2 \sqsubseteq b_2 \\ d_2 \sqsubseteq c_2 & e_2 \sqsubseteq c_2 & f_2 \sqsubseteq b_2 \\ d_2 \sqsubseteq a_2 & e_2 \sqsubseteq a_2 & f_2 \sqsubseteq a_2 \\ g_2 \sqsubseteq a_2 & & \end{array} \right. \\
 Cn_{\Omega, \Lambda}^{\omega}(o_3) &= \left\{ \begin{array}{lll} b_3 \sqsubseteq a_3 & c_3 \sqsubseteq a_3 & g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3 & e_3 \sqsubseteq c_3 & f_3 \sqsubseteq b_3 \\ i \in e_3 & b_3 \perp c_3 & f_3 \sqsubseteq a_3 \quad g_3 \sqsubseteq a_3 \\ d_3 \sqsubseteq a_3 & e_3 \sqsubseteq a_3 & i \in c_3 \\ i \in a_3 & d_3 \perp b_3 & e_3 \perp b_3 \\ f_3 \perp c_3 & g_3 \perp c_3 & b_3 \supseteq e_3 \\ i \in b_3 & & \end{array} \right.
 \end{aligned}$$

together with three alignments $A_{1,2}$, $A_{2,3}$, and $A_{3,1}$. These alignments can be described as

follows:

$$\begin{aligned}
 Cn_{\Omega,\Lambda}^{\alpha}(o_1, o_2) &= \begin{cases} b_1 \leq d_2 & b_1 \leq c_2 \\ b_1 \leq a_2 \end{cases} \\
 Cn_{\Omega,\Lambda}^{\alpha}(o_2, o_3) &= \begin{cases} c_2 \leq b_3 & c_2 \leq a_3 \\ d_2 \leq b_3 & e_2 \leq b_3 \\ d_2 \leq a_3 & e_2 \leq a_3 \\ c_2 \perp c_3 \\ d_2 \perp c_3 & e_2 \perp c_3 \\ d_2 \perp d_3 & e_2 \perp d_3 \\ d_2 \perp e_3 & e_2 \perp e_3 \\ d_2 \geq e_3 & c_2 \geq e_3 \\ a_2 \geq e_3 \end{cases} \\
 Cn_{\Omega,\Lambda}^{\alpha}(o_1, o_3) &= \begin{cases} e_1 \geq f_3 & b_1 \geq e_3 \\ c_1 \geq f_3 & a_1 \geq f_3 \\ a_1 \leq e_3 & b_1 \leq b_3 \\ b_1 \leq a_3 & b_1 \perp c_3 \\ b_1 \perp d_3 & b_1 \perp e_3 \end{cases}
 \end{aligned}$$

A more semantic definition of subsumption could be such that $h(o) \models_{\Omega,\Lambda} o$ and $k(A) \models_{\Omega,\Lambda} A$, i.e., $\forall o \in \Omega, Cn_{\Omega,\Lambda}^{\omega}(o) \subseteq Cn_{\Omega',\Lambda'}^{\omega}(h(o))$ and $\forall A \in \Lambda, Cn_{\Omega,\Lambda}^{\alpha}(A) \subseteq Cn_{\Omega',\Lambda'}^{\alpha}(k(A))$.

Example 42 (Inconsistency). A model of the network of ontologies presented in Example 29, is a triple of models $\langle m_1, m_2, m_3 \rangle$ satisfying all alignments, i.e., in particular, satisfying:

$$\begin{aligned}
 m_3(b_3) &\supseteq m_2(c_2) \\
 m_2(c_2) &\supseteq m_1(b_1) \\
 m_1(b_1) &\supseteq m_3(e_3)
 \end{aligned}$$

But, it is clear that all models m_3 of o_3 must satisfy $m_3(b_3) \cap m_3(c_3) = \emptyset$, $m_3(i) \in m_3(e_3)$, and $m_3(i) \in m_3(c_3)$. Moreover, all models m_2 of o_2 must satisfy $m_2(d_2) \subseteq m_2(c_2)$. Hence, $m_3(b_3) \supseteq m_1(b_1)$, $m_3(b_3) \supseteq m_3(e_3)$ and then $m_3(i) \in m_3(b_3)$, which is contradictory with previous assertions. So there cannot exist such a triple of models and there is no model for this network of ontologies.

In this example, taking any of the ontologies with only the alignments which involve them, e.g., $\langle \Omega, \{A_{1,3}, A_{2,3}\} \rangle$, is a consistent network of ontologies.

The closure of a network of alignments may introduce non empty alignments between ontologies which were not previously connected or empty. This is possible because constraints do not come locally from the alignment but from the whole network of ontologies. Such a representation is highly redundant as closures usually are.

Exercise: prove that $Cn(\langle \Omega, \Lambda \rangle)$ is semantically equivalent to $\langle \Omega, \Lambda \rangle$.

11.2 Local models from the standpoint of an ontology

This definition coincides with a coherent model of the world in which all models satisfy all alignments. This is the standpoint of an omniscient observer and it corresponds to the global knowledge of a distributed system as defined in [Fagin *et al.*, 1995].

However, if one agent has an inconsistent ontology then the network of ontologies has no model. Therefore, even agents not connected to the inconsistent ontology cannot compute reasonable models. Moreover, an agent knowing an ontology and the related alignments would like to use the system by gathering information from its neighbours and considering only the models of this information. Thereby, it would be able to compute consequence through some complete deduction mechanisms. This is important when asking agents to answer queries and corresponds to local knowledge in [Fagin *et al.*, 1995]. This is the knowledge an agent can achieve by communicating only with the agents it is connected to in a distributed system (see below).

From that standpoint, there can be several ways to select the acceptable models given the distributed system:

$$\begin{aligned}
 \mathcal{M}_{\Omega,\Lambda}^0(o) &= \mathcal{M}(o) \\
 \mathcal{M}_{\Omega,\Lambda}^1(o) &= \{m \in \mathcal{M}(o); \forall o' \in \Omega, \exists m' \in \mathcal{M}(o'); m, m' \models \lambda(o, o')\} \\
 \mathcal{M}_{\Omega,\Lambda}^2(o) &= \{m \in \mathcal{M}(o); \forall o' \in \Omega, \exists m' \in \mathcal{M}_{\Omega,\Lambda}^2(o'); m, m' \models \lambda(o, o')\} \\
 \mathcal{M}_{\Omega,\Lambda}^3(o) &= \{m \in \mathcal{M}(o); \forall o' \in \Omega, \exists \vec{m} \in \mathcal{M}(\langle \Omega, \Lambda \rangle); \vec{m}_o = m \wedge m, \vec{m}_{o'} \models \lambda(o, o')\} \\
 \mathcal{M}_{\Omega,\Lambda}^5(o) &= \{m \in \mathcal{M}(o); \forall o' \in \Omega, \forall m' \in \mathcal{M}_{\Omega,\Lambda}^5(o'); m, m' \models \lambda(o, o')\} \\
 \mathcal{M}_{\Omega,\Lambda}^6(o) &= \{m \in \mathcal{M}(o); \forall o' \in \Omega, \forall m' \in \mathcal{M}(o'); m, m' \models \lambda(o, o')\} \\
 \mathcal{M}_{\Omega,\Lambda}^\forall(o) &= \{m \in \mathcal{M}(o); \forall o' \in \Omega, \forall \vec{m} \in \mathcal{M}(\langle \Omega, \Lambda \rangle); \vec{m}_o = m \wedge m, \vec{m}_{o'} \models \lambda(o, o')\}
 \end{aligned}$$

These approaches have been ordered from the more optimistic to the more cautious. $\mathcal{M}_{\Omega,\Lambda}^1$ selects the models that satisfy each alignment in at least one model of the connected ontology. $\mathcal{M}_{\Omega,\Lambda}^6$ is very strong since all alignments must be satisfied by all models of the connected ontologies. $\mathcal{M}_{\Omega,\Lambda}^2$ and $\mathcal{M}_{\Omega,\Lambda}^5$ are fixed point characterisations that, instead of considering the initial models of the connected agents, consider their selected models by the same function. This contributes propagating the constraints to the whole connected components of the network of ontologies. While for $\mathcal{M}_{\Omega,\Lambda}^2$ this strengthens the constraints, for $\mathcal{M}_{\Omega,\Lambda}^5$, this relaxes them with respect to $\mathcal{M}_{\Omega,\Lambda}^6$. Here, an inconsistent model is a problem only to related agents and only for versions $\mathcal{M}_{\Omega,\Lambda}^1$, and $\mathcal{M}_{\Omega,\Lambda}^2$ which require the existence of a model for each related ontology.

Each of these options allows the definition of a semantics for network of ontologies that is different from the model of network of ontologies considered above. It is also analogous to the distributed knowledge of the system following [Fagin *et al.*, 1995].

One can be even more restrictive, as in local model semantics [Ghidini and Giunchiglia, 2001], by considering only a subset of the possible models of each ontology.

When dealing with ontology matching between a pair of ontologies, the matter of semantics between ontologies is not related to the alignment but to the interpretation of the full network of ontologies, for instance, depending if one wants to enforce global consistency or not. In this book we will not take a position on such a matter and will only retain the basic interpretation framework provided above.

11.3 Conclusion

Having a semantics for alignments allows for defining what is entailed by putting ontologies together. So it contributes giving a meaning to the semantic web.

In addition, the definition of consequence operations on ontologies or alignments may be used for reasoning with these. For instance, it is possible to formally define alignment composition [Zimmermann and Euzenat, 2006; Zimmermann, 2008]. Alignment composition syntactically computes a subset of α -consequences. It may even be used for reasoning about networks of ontologies without knowing the ontologies themselves. More precise reasoning may help testing for alignment consistency which, in turn, may be used for improving matcher results. We have shown that this semantics may also be used for defining semantic evaluation measures for alignments provided by matchers [Euzenat, 2007].

A network of ontology can be seen as an ontology defining a \models relation. It thus can be included within a network of ontologies as an ontology. This model relation may also be used in the definition of query answering from the standpoint of a particular ontology, hence:

$$A_{\Omega, \Lambda}(\vec{B}, o, P) = \{\sigma |_{\vec{B}} | o \models_{\Omega, \Lambda} \sigma(P)\}$$

In fact, ontologies and alignments may be thought of as complementary: it is possible to use alignments for completing ontologies and using ontologies for completing alignments (while satisfying the semantics).

Finally, such a formalism contributes to the definition of the meaning of alignments: it describes what are the consequences of ontologies with alignments, i.e., what can be deduced by an agent. However, it does not describe what the correct alignments are: matching is not a deductive task but an inductive one. The framework is nevertheless particularly useful for deciding if delivered alignments are consistent, i.e., if networks of ontologies have a model or not. Hence, it is useful for specifying what is expected from matching algorithms and how they should be designed or evaluated.

11.4 Exercises

Exercise 13 (Network of ontologies). *In addition of the ontology o of Exercise 7, we consider an ontology o' which defines the class $op: Buch$ and contains the following statements:*

$\langle d: Baudelaire, o: translated, d: Confessions \rangle \langle d: DeQuincey, o: wrote, d: Confessions \rangle$

and o'' which defines the class $opp: Roman$ and contain the following statements:

$\langle d: Confessions, rdf: type, opp: Roman \rangle \langle d: Musset, o: translated, d: Confessions \rangle$

They are related together by the following three alignments:

- $A_{o, o'} = \{ \langle o: Literature, \equiv, op: Buch \rangle \}$
- $A_{o', o''} = \{ \langle op: Buch, \sqsubseteq, opp: Roman \rangle \}$
- $A_{o'', o} = \{ \langle opp: Roman, \equiv, o: Novel \rangle \}$

So that we have a network of ontology $\langle \{o, o', o''\}, \{A_{o, o'}, A_{o', o''}, A_{o'', o}\} \rangle$.

1. *Do you think that this network of ontologies is well designed? Why?*
2. *Is this network consistent? Provide a model for this network of ontologies.*
3. *Provide the constraints that the alignments impose on models.*
4. *What does this entail for the class $(rdf: type)$ of $d: Confessions$ and $d: TheRaven$ at o in this network?*

Chapter 12

Equalising semantics for alignments

The usual way of providing a semantics for related conceptual systems is through modal logic of knowledge and belief [Fagin *et al.*, 1995; Wooldridge, 2000]. In the line of the work on data integration, we only give 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 terms of the semantics of each individual ontology.

The main problem arising is the non compatibility of the domains of interpretation. Given several ontologies, it is possible to consider different positions with regard to the domain of interpretation:

- For all these ontologies, the domain of interpretation D is unique. This approach is useful when ontologies describe a set of well defined entities, like the set of files shared in a peer-to-peer system. This approach has been taken in [Calvanese *et al.*, 2002; Calvanese *et al.*, 2004].
- For each ontology o , the domain D_o may be different. Domains are related with the help of domain relations $r_{o,o'}$ which map elements of D_o to corresponding elements of domain $D_{o'}$. This approach is used in [Ghidini and Serafini, 1998; Borgida and Serafini, 2003].
- There is no constraint on the domain of interpretation of ontologies. This is the assumption that will be considered here. For dealing with this assumption, we use a universal domain U , that may be defined as the union of all the domains under consideration, and an equalising function γ or rather a set of equalising functions: $\gamma_o : D_o \rightarrow U$.

[Zimmermann and Euzenat, 2006] considers the implications of these three models. Here,



Figure 12.1: Equalising semantics.

because the models of various ontologies can have different interpretation domains, we use the notion of an equalising function, which helps make these domains commensurate.

Definition 62 (Equalising function). *Given a family of interpretations $\langle I_o, D_o \rangle_{o \in \Omega}$ of a set of ontologies Ω , an equalising function for $\langle I_o, D_o \rangle_{o \in \Omega}$ is a family of functions $\gamma = (\gamma_o : D_o \rightarrow U)_{o \in \Omega}$ from the ontology 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 equalising functions can be different for each ontology. This means, in particular, that even if two ontologies are interpreted over the same domain of interpretation, it is not compulsory that the equalising function maps their elements to the same element of U , though it remains possible. This allows for a loose coupling of the interpretations.

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 63 (Interpretation of alignment relations). *Given $r \in \Theta$ 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 64 (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*

$$\langle \gamma_o(m(e)), \gamma_{o'}(m'(e')) \rangle \in r^U$$

This is denoted as $m, m' \models_\gamma c$.

This definition may be reduced to definition Definition 57 if either γ is constrained to be the identity function or $\gamma_o \cdot m \in \mathcal{M}(o)$, $\gamma_{o'} \cdot m' \in \mathcal{M}(o')$.

Definition 65 (Satisfied alignment). *An alignment A is satisfied for an equalising function γ by two models m, m' of o, o' if and only if all its correspondences are satisfied for γ by m and m' . This is denoted as $m, m' \models_\gamma A$.*

This is useful for defining the classical notions of validity and satisfiability.

Definition 66 (Alignment validity). *An alignment A of two ontologies o and o' is said to be valid if and only if*

$$\forall m \in \mathcal{M}(o), \forall m' \in \mathcal{M}(o'), \forall \gamma \in \Gamma, m, m' \models_\gamma A$$

This is denoted as $\models A$.

From the practical perspective, this is not a very useful definition since unless the ontologies have no models, it will be very difficult to find valid alignments. A relaxed definition could consider the validity that, given an equalising function, describes how domains are related. Valid alignments are direct logical consequences of the two ontologies. Thus they do not provide additional information than what is already in these ontologies. Satisfiable alignments offer more information.

Definition 67 (Satisfiable alignment). *An alignment A of two ontologies o and o' is said to be satisfiable if and only if*

$$\exists m \in \mathcal{M}(o), \exists m' \in \mathcal{M}(o'), \exists \gamma \in \Gamma; m, m' \models_{\gamma} A$$

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. The satisfiable set of alignments is far larger than the set of valid ones. Again, one can define γ -satisfiable alignments, i.e., alignments satisfiable for a given equalising function.

Given an alignment between two ontologies, the semantics of the aligned ontologies can be defined as follows.

Definition 68 (Models of aligned ontologies). *Given two ontologies o and o' and an alignment A between these ontologies, a model m'' of these ontologies aligned by A is a pair $\langle m, m' \rangle \in \mathcal{M}(o) \times \mathcal{M}(o')$ such that there exists $\gamma \in \Gamma$, such that $m, m' \models_{\gamma} A$.*

This provides the necessary definitions for defining a models of networks of ontologies.

Definition 69 (Models of networks of ontologies). *Given a finite set of n ontologies Ω and a finite set of alignments Λ between pairs of ontologies in Ω , a model of the network of ontologies $\langle \Omega, \Lambda \rangle$ is a n -uple of models $\langle m_1 \dots m_n \rangle \in \mathcal{M}(o_1) \times \dots \mathcal{M}(o_n)$, such that there exists $\gamma \in \Gamma$, such that for each alignment $A \in \Lambda(o_i, o_j)$, A is satisfied by $\langle m_i, m_j, \gamma \rangle$.*

A definition of acceptable models for an ontology, corresponding to a refinement $\mathcal{M}_{\Omega, \Lambda}^2$ in which the equalising function appears, is given as follows:

Definition 70 (Models of an ontology modulo alignments). *Given a network of ontologies $\langle \Omega, \Lambda \rangle$, the models of $o \in \Omega$ modulo Λ are those models of o , such that for each ontology o' there exists a model that satisfies all elements of Λ between o and o' :*

$$\mathcal{M}_{\Omega, \Lambda}^4(o) = \{m \in \mathcal{M}(o); \forall o' \in \Omega, \exists \gamma \in \Gamma; \exists m' \in \mathcal{M}_{\Omega, \Lambda}^4(o'); m, m' \models_{\gamma} \lambda(o, o')\}$$

Chapter 13

Application: Distributed query evaluation

Networks of ontologies allow for connecting ontologies together in a meaningful way. Hence, they can help interpreting information from one ontology into another ontology. In practice, this may be performed in many different ways. Here we illustrate this in a particular type of systems: semantic peer-to-peer systems.

13.1 Semantic peer-to-peer systems

The semantic web is a set of data sources providing information either by answering queries or by offering RDF graphs or snippets. These may not be networks of ontologies because, they may share the same ontologies and not the same data. They are more alike what we call a semantic peer-to-peer network.

A peer-to-peer sharing network is a set of nodes each one aware of other nodes called its acquaintances able to share some resources, e.g., documents. They become semantic when their resources are described through semantic web technologies.

Figure 13.1 illustrates such a system.

It is more formally defined in Definition 71. This definition is very close to other definitions (usually not using the concept of network of ontologies), such as the one developed in [Cudré-Mauroux, 2008].

Definition 71 (Semantic peer-to-peer system). *A semantic peer-to-peer system is a tuple $\langle \Pi, \Omega, \Lambda, \Psi, \alpha, \omega, \psi, \rho \rangle$, such that:*

- Π is a set of peers;
- $\langle \Omega, \Lambda \rangle$ is a normalised network of ontologies;
- Ψ is a set of resources identified by their URIs disjoint from those of Ω ;
- $\alpha : \Pi \rightarrow 2^\Pi$ is a function associating to each peer its acquaintances;
- $\omega : \Pi \rightarrow \Omega$ is a function associating to each peer its ontology;
- $\psi : \Pi \rightarrow 2^\Psi$ is a function associating to a peer the resources it knows;
- $\rho : \Pi \rightarrow RDF$ is a function associating to each peer its data as an RDF graph made out of $(\psi(p) \times \{rdf : type\} \times \omega(p)) \cup (\psi(p) \times \omega(p) \times \psi(p))$

It is possible to complexify this schema by having a peer using several ontologies at once ($\omega : \Pi \rightarrow 2^\Omega$) and/or by considering that each peer knows a different part of the network of

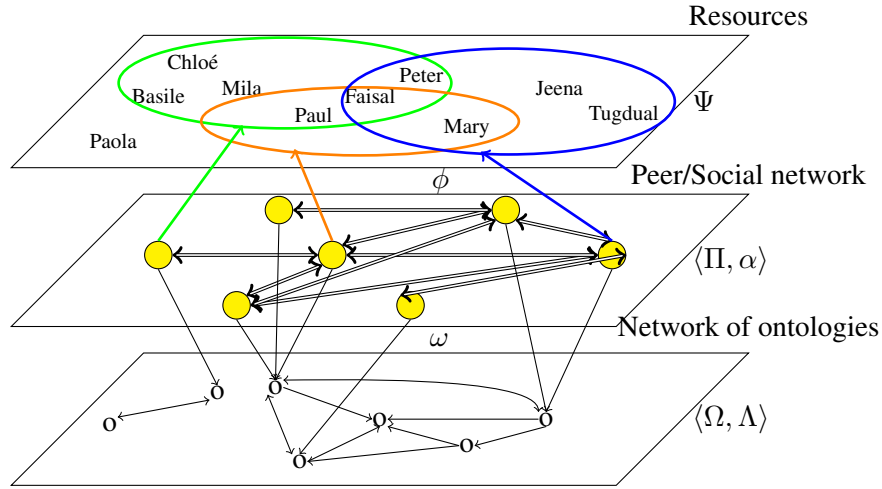


Figure 13.1: A semantic peer-to-peer system.

ontologies (it is not public: $\eta : \Pi \rightarrow 2^\Omega \times 2^\Lambda$). But, in first approximation, we only consider Definition 71. There is something more about a peer-to-peer semantic system which is the way to acquire knowledge, e.g., its query language. However, we will not consider here.

Now that we know what is the semantics of networks of ontologies, the semantics of an RDF graph and that of SPARQL query, how can we define the semantics of a semantic peer-to-peer system?

13.2 The semantics of semantic peer-to-peer systems

With respect to the previous semantics, there is a common dataset, Ψ which facilitates the expression of the semantics because peers are only interested in interpretations for which the domain is (congruent to) Ψ .

13.2.1 What the system knows?

The system has absolute knowledge. It has access to all ontologies and all data graphs in the system. Hence it can put all these graphs together and what it knows is:

$$\bigcup_{p \in \Pi} \{ \delta; \omega(p), \rho(p) \models_{\Omega, \Lambda} \delta \}$$

which should correspond to the consequences of the network of ontology amplified by each peer's data graph:

$$Cn(\langle \{ o \in \Omega; o \cup \bigcup_{p \in \Pi; \omega(p)=o} \rho(p) \}, \Lambda \rangle)$$

It is noteworthy that this is independent from the topology of the peer-to-peer system (the acquaintances).

13.2.2 What a peer knows by itself?

What a peer knows in isolation depends only on its ontology and data graph:

$$\{\delta \in \psi(p); \omega(p), \rho(p) \models \delta\}$$

However, since the network of ontologies is public, it can be used for having more information:

$$\{\delta \in \psi(p); \omega(p), \rho(p) \models_{\Omega, \Lambda} \delta\}$$

So, if a peer is seeking for the answer to a query P , it will be:

$$\mathcal{A}^\pi(\vec{B}, p, P) = \mathcal{A}_{\Omega, \Lambda}(\vec{B}, \omega(p) \cup \rho(p), P)$$

13.2.3 What a peer will (eventually) know?

The goal of peer-to-peer systems is to take advantage of other peer's knowledge for gathering more information. The peer is limited by its acquaintances in addition to the connectivity of the network of ontologies. In particular, given a particular topology of the peer network, it can only acquire information which are accessible to him.

It can acquire information about resources that it did not know and do it based on ontologies it does not master. The answer to the same query in this context will be:

$$\mathcal{A}^{\pi*}(\vec{B}, p, P) = \mathcal{A}^\pi(\vec{B}, p, P) \cup \bigcup_{q \in \alpha(p)} \mathcal{A}^{\pi*}(\vec{B}, q, \lambda_{\omega(p), \omega(q)}(P))$$

It is limited by a non introduced element: the query language used for expressing P . This definition also uses an undefined notation: the application of an alignment to a query. It assumes here that the query may be translated by the alignment. This is not always the case, so, in first approximation, we assume that the answer is empty if the translation is impossible.

However, peers may develop strategies in order to decompose queries and send fragment of these to peers which can deal with these fragments, i.e., such that $\lambda_{\omega(p), \omega(q)}$ is defined on these fragments (see Figure 13.2). Peer may also only send the query to a restricted number of peers if they are confident that the gain in time outweighs the (eventual) loss in completeness.

What a peer may know may not necessarily reach what the system knows. That's life.

13.3 Exercises

Exercise 14 (Semantic peer-to-peer system). *Given a peer-to-peer system with peers n_1 , n_2 and n_3 all related to each others¹. Peers n_1 and n_2 share the ontology o that has been defined at Question 1 of Exercise 4, n_3 uses the ontology o' defining;*

```
t:Work rdf:type owl:Class.
t:copyrightHolder rdf:type rdf:Property.
t:copyrightHolder rdfs:domain t:Work.
t:year rdf:type rdf:Property.
t:year rdfs:domain t:Work.
t:year rdfs:range xsd:integer.
```

¹This exercise has not been given at the actual exam.

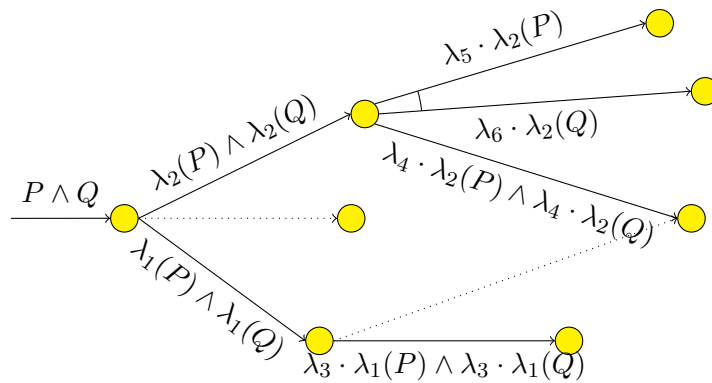


Figure 13.2: Query evaluation strategy.

There exists an alignment A between o and o' containing two correspondences: $t:\text{Work} \geq m:\text{Livre}$, $t:\text{year} \equiv m:\text{annee}$. Assume that n_1 does not contain instances; n_2 contains:

```
http://mm.com#a345 m:aecrit http://isbn.org/2070360423.
http://mm.com#a345 m:aecrit http://isbn.org/2070360024.
http://mm.com#a345 m:aecrit http://isbn.org/2070322882.
http://mm.com#a345 foaf:name "Albert".
http://isbn.org/2070360423 rdf:type m:Roman.
http://isbn.org/2070360423 dc:title "La peste".
http://isbn.org/2070360024 rdf:type m:Roman.
http://isbn.org/2070360024 dc:title "L'Étranger".
http://isbn.org/2070322882 rdf:type m:Livre.
http://isbn.org/2070322882 dc:title "Le Mythe de Sisyphe".
...
```

and n_3 contains:

```
http://isbn.org/2070360423 t:year 1947.
http://isbn.org/2070322882 t:year 1942.
...
```

1. Express the alignment A in OWL.
2. The peer n_1 would like to evaluate the following query:

```
SELECT ?t, ?y
PREFIX ...
WHERE
  ?x foaf:name "Albert".
  ?x m:aecrit ?l.
  ?l rdf:type m:Roman.
  ?l dc:title ?t.
  OPTIONAL { ?l m:annee ?y. }
```

How is it possible to answer this query by using n_2 , n_3 and A ?

3. Provide the answer on the available data.

	P2P	PDMS	SW
$\rho(p)$	membership	+relations	RDF
Ω	terminologies	DL-Lite	OWL
Q_L	classes	queries	EDOAL
Θ	equivalence	subsumption	relation algebra
queries	class	conjunctive queries	SPARQL _{OWL}

Table 13.1: Different expressivity choices for (semantic) peer-to-peer systems (any combination is possible).

13.4 Conclusions

Semantic peer-to-peer systems have been introduced as an application of the concepts which have been considered in this lecture.

It is a very flexible definition. Depending on the languages retained for expressing graphs, ontologies or for transmitting queries between peers one obtains a peer-to-peer system with very different properties ranging from classical peer-to-peer systems to full semantic web technologies (see Table 13.1). We have attempted to show the flexibility of the semantic web technologies, so that simply replacing one language by another does not change the definitions (and hopefully the technology for implementing such systems).

This definition remains static: we consider one semantic peer-to-peer network disregarding its evolution. What characterises such systems is that they occur in time: peers acquire information that they include in their knowledge base (ρ), or meet new acquaintances (α); new peers appears and old ones disconnect (Π); and even ontologies (Ω) and alignments (Λ) evolve. Since, any element of the model may evolve over time, it is useful to be able to track these evolutions and to know how the network, or an individual peer, is supposed to react to such changes.

But, this is another story.

Chapter 14

Conclusion

These lectures led us to define in a very precise way what could be the semantics of the semantic web. The semantic web is made of independent and heterogeneous sources of information available to anyone. We have precised the syntax of *graph* formalisms in which this information is offered and have provided a semantics for this syntax. Then we have considered the definition of the vocabulary (*ontology*) used for expressing this information and again provided the semantics of graphs using this vocabulary language. We have also defined a way to extract information from these sources through *queries* and again, we have defined the syntax and semantics of such queries. Finally, because the sources of information do not depend on the same ontologies we have introduced *alignments* for expressing the relationships between ontologies, and defined their syntax and semantics.

The final example showed that these elements can be put together for designing and describing operating systems.

From there further questions may be asked, such as:

- How to model different styles of systems using the same resources?
- What is the impact of query or ontology languages on what may be known by a peer?
- What is the complexity of answering one type of query in such a system?
- How to design an efficient strategy for extracting as much information as possible?
- How to remain robust to inconsistent ontologies or peers?

All these are research questions. . .

Part IV

Correction of exercises

RDF

Exercise 1 (RDF assertions). Consider the four graphs of Figure 2.5.

1. Are they all well-formed RDF graphs? Why?

Graph (b) is not a well-formed RDF graph because a literal, "La peste" is the subject of a relation (`rdf:type`).

2. Express the graph of Figure 2.5(a) as a set of triples.

```
?x foaf:name "Albert". ?x m:acrit ?l. ?l rdf:type m:Roman. ?l
dc:title "La peste".
```

You will list the sets of literals, URIRefs and variables (or blanks) found in this graph.

Literals: "Albert", "La peste"

URIRefs: `foaf:name`, `m:acrit`, `m:Roman`, `rdf:type`, `dc:title`

Variables: `?x`, `?l`

3. Give an informal meaning of this graph or its expression in the predicate calculus

Some entity named "Albert" has written a Novel titled "La peste".

$$\exists x, \exists l, \text{name}(x, \text{"Albert"}) \wedge \text{acrit}(x, l) \wedge \text{Roman}(l) \wedge \text{title}(l, \text{"Lapeste"})$$

4. Consider the RDF graphs of Figure 2.5 which are well-formed, do some of them entail others?

$$(a) \models (d) \text{ and } (d) \models (a)$$

Explain why.

Since (a) is a strict subgraph of (d), then it is obvious that $(d) \models (a)$.

For the opposite direction, it is sufficient to see that the node labeled by `?y` in (d) can be projected to the node labeled `?l` in (a) while projecting the other nodes to those nodes bearing the same label. This projection preserves all the edges of the graph and it is complete, then $(a) \models (d)$.

In predicate calculus, this is:

$$\exists x, \exists l, \text{name}(x, \text{"Albert"}) \wedge \text{acrit}(x, l) \wedge \text{Roman}(l) \wedge \text{title}(l, \text{"La peste"})$$

is equivalent to:

$$\exists x, \exists l, \text{name}(x, \text{"Albert"}) \wedge \text{acrit}(x, l) \wedge \text{Roman}(l) \wedge \text{title}(l, \text{"La peste"}) \wedge \exists y, \text{acrit}(x, y)$$

which is equivalent to:

$$\exists x, \exists l, \exists y, \text{name}(x, \text{"Albert"}) \wedge \text{acrit}(x, l) \wedge \text{Roman}(l) \wedge \text{title}(l, \text{"La peste"}) \wedge \text{acrit}(x, y)$$

Exercise 2 (RDF and assertions). Consider the two graphs of Figure 2.6 dealing with the expression of social relationships.

-
1. Express the graph of Figure 2.6(b) as a set of triples. You will give the list of literals, URIRefs and variables (or blanks) in this graph.

$$\langle ?a, foaf:mbox, "paul@e.at" \rangle \langle ?a, worksWith, ?b \rangle$$

$$\langle ?a, playsTennisWith, ?c \rangle \langle ?b, marriedWith, ?c \rangle$$

$$\langle ?b, foaf:mbox, "keith@g.es" \rangle \langle ?b, foaf:mbox, "keith@i.com" \rangle$$

$$\mathcal{L} = \{ "paul@e.at", "keith@i.com", "keith@g.es" \}$$

$$\mathcal{U} = \{ worksWith, playsTennisWith, foaf:mbox, marriedWith \}$$

$$\mathcal{B} = \{ ?a, ?b, ?c \}$$

2. What is, informally, the meaning of the graph of Figure 2.6(b) (tell it in English or French or predicate calculus)?

An individual whose email address is "paul@e.at" plays tennis with the spouse of a colleague whose email addresses are "keith@g.es" and "keith@i.com". Or, in predicate calculus:

$$\exists a, \exists b, \exists c; worksWith(a, b) \wedge playsTennisWith(a, c) \wedge marriedWith(b, c)$$

$$\wedge mbox(a, "paul@e.at") \wedge mbox(b, "keith@g.es") \wedge mbox(b, "keith@i.com")$$

3. Does one of these graphs entail the other? (explain why)

$a \not\models b$ because it is not possible to deduce that $\exists x; x marriedWith c$ and $x foaf:mbox keith@i.com$

$b \not\models a$ because it is not possible to deduce $worksWith$ from $foaf:knows$.

Another interesting answer has been given by a student. The following SPARQL query:

```
ASK ?x worksWith ?y. ?y foaf:mbox "keith@i.com".
```

Succeed for (b) and fails for (a), while

```
ASK ?x foaf:knows ?y.
```

succeed for (a) and fails for (b). Hence, none of these graphs entail the other.

Exercise 3 (RDF graphs). Here are the 8 triples of an RDF graph G about writers and their works: (all identifiers correspond in fact to URIs, $_:b$ is a blank node):

$$\langle d:Poe, o:wrote, d:TheGoldBug \rangle \langle d:Baudelaire, o:translated, d:TheGoldBug \rangle$$

$$\langle d:Poe, o:wrote, d:TheRaven \rangle \langle d:Mallarmé, o:translated, d:TheRaven \rangle$$

$$\langle d:TheRaven, rdf:type, o:Poem \rangle \langle d:Mallarmé, o:wrote, _:b \rangle$$

$$\langle _:b, rdf:type, o:Poem \rangle \langle d:TheGoldBug, rdf:type, o:Novel \rangle$$

1. Draw an RDF graph corresponding to these statements
2. Express in English the meaning of these statements.

Poe wrote the Poem "The Raven" translated by Mallarmé and the novel "The Gold Bug" translated by Baudelaire. Mallarmé wrote a poem.

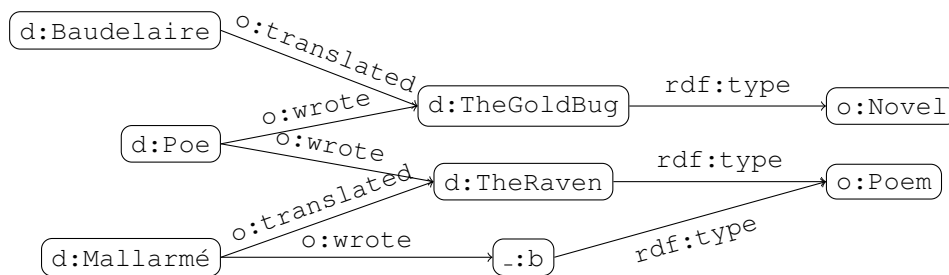


Figure 14.1: The RDF graph G .

OWL

Exercise 4 (OWL ontologies).

1. Describe in OWL (RDF or XML/RDF) the ontology containing the following assertions:

- All authors are persons;
- A book ($m:livre$) has exactly one year of publication ($m:annee$);
- A novel ($m:roman$) is a book ($m:livre$) and a book is a work ($m:oeuvre$);
- The title ($dc:title$) of a work is a character string ($xsd:string$);
- The relation "a écrit" ($m:aecrit$) relates an author to a work.

```

<owl:Class rdf:about="#Auteur"> <rdfs:subClassOf
rdf:resource="#Personne"/> </owl:Class>
<owl:Class rdf:about="#Oeuvre"> <rdfs:subClassOf>
<owl:Restriction> <owl:onProperty rdf:resource="dc:title"/>
<owl:datatype>&xsd:string</owl:datatype> </owl:Restriction>
</rdfs:subClassOf> </owl:Class>
<owl:Class rdf:about="#Livre"> <rdfs:subClassOf
rdf:resource="#Oeuvre"/> <rdfs:subClassOf>
<owl:Restriction> <owl:onProperty rdf:resource="#annee"/>
<owl:cardinality>1</owl:cardinality> </owl:Restriction>
</rdfs:subClassOf> </owl:Class>
<owl:Class rdf:about="#Roman"> <rdfs:subClassOf
rdf:resource="#Livre"/> </owl:Class>
<owl:ObjectProperty rdf:about="#aecrit"> <rdfs:domain
rdf:resource="#Auteur"/> <rdfs:range rdf:resource="#Oeuvre"/>
</owl:ObjectProperty>
  
```

which corresponds to the diagram of Figure ??.

2. If one associates the graph (a) of Figure 2.5 of Exercise 1 (p20) and the ontology resulting from the previous question, is it possible to deduce the type ($rdf:type$) of $?x$?

$?x \text{ rdf:type } m:Auteur.$

Can you semantically justify how?

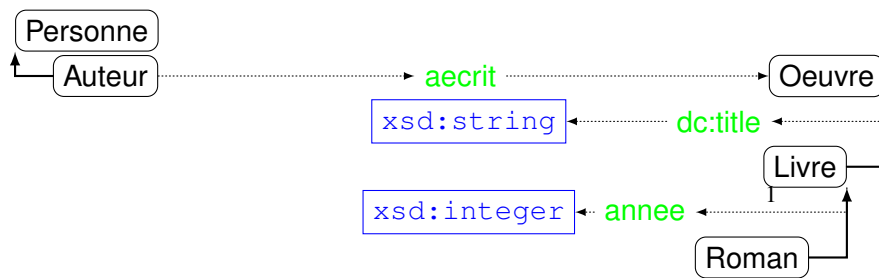


Figure 14.2: OWL ontologies.

$I(aecrit) \subseteq I(Auteur) \times I(Oeuvre)$

$\langle I(?x), I(?l) \rangle \in I(aecrit)$

hence

$I(?x) \in I(Auteur)$

thus: $?x \text{ rdf:type } Auteur$ What else can be deduced?

$?l$ has a year of publication: $?l \text{ m:annee } ?y$.

3. Does the use of this ontology for defining the graphs of Figure 2.5 (p20) would change something to the answer to Question 3 of Exercise 1? What?

Yes: (a) \models (c) because $?l \text{ rdf:type } m:Roman \models ?l \text{ rdf:type } m:Livre$

and (d) \models (c) because (d) \models (a).

Exercise 5 (DL-Lite ontologies). Consider the ontology O made of the following DL-Lite assertions:

```
worksWith  $\sqsubseteq$  foaf:knows
playsTennisWith  $\sqsubseteq$  playsSportWith
playsSportWith  $\sqsubseteq$  foaf:knows
marriedWith  $\sqsubseteq$  foaf:knows
Person  $\sqsubseteq$   $\exists$ foaf:mbox
```

1. In which dialect (sublanguage) of DL-Lite is this ontology expressed (DL-Lite_{core}, DL-Lite_F, DL-Lite_R)?

This is DL-Lite_R because there are assertions on relations.

2. Rewrite O in OWL or RDFS.

```
<owl:ObjectProperty rdf:about="worksWith">
  <rdfs:subPropertyOf rdf:resource="#foaf;knows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="playsTennisWith">
  <rdfs:subPropertyOf rdf:resource="#playsSportWith" />
</owl:ObjectProperty>
```

```

<owl:ObjectProperty rdf:about="playsSportWith">
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="marriedWith">
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:Class rdf:about="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox" />
      <owl:someValuesFrom rdf:resource="&owl;Thing" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

or, for the last one:

```

<owl:Class rdf:about="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

3. In which dialect (sublanguage) of RDFS or OWL is your ontology expressed?

This is *OWL-Lite*. Although, $DL\text{-}Lite_{\mathcal{R}}$ is more expressive than RDFS and non comparable with *OWL-Lite*, this particular ontology is expressed in *OWL-Lite* (in which *minCardinality* constraints are expressible with values of 0 or 1).

4. Given the graph of Figure 2.6(b) of Exercise 2 (p21) to which the axioms of *O* are added. Compute its closure. What is the difference with the partial closure?

The closure is:

```

⟨?a, foaf:mbox, "paul@e.at"⟩⟨?a, worksWith, ?b⟩
  ⟨?a, playsTennisWith, ?c⟩⟨?b, marriedWith, ?c⟩
⟨?b, foaf:mbox, "keith@g.es"⟩⟨?b, foaf:mbox, "keith@i.com"⟩
  ⟨?a, foaf:knows, ?c⟩⟨?a, foaf:knows, ?b⟩
  ⟨?a, playsSportWith, ?c⟩⟨?b, foaf:knows, ?c⟩

```

plus the axiomatic triples (like $\langle foaf:mbox, rdf:type, rdf:Property \rangle$, $\langle rdf:type, rdf:type, rdf:Property \rangle$, $\langle rdf:_1, rdf:type, rdf:Property \rangle \dots$). The partial closure is the same minus the triples involving $rdf:i$ because there is no such URI in our graph.

-
5. Given the graphs of Figure 2.6 (p21) to which the axioms of O are added, does this change something to the answers given to Question 3 of Exercise 2 if we consider RDFS-entailment? (explain why)

Now $b, O \models a$ because *worksWith* \sqsubseteq *foaf:knows*.

Exercise 6 (OWL ontologies).

1. Provide an ontology satisfied by both graphs of Figure 2.6 of Exercise 2 (p21).

```
<owl:ObjectProperty rdf:about="&foaf;knows">
<rdfs:domain rdf:resource="&foaf;Person" /> <rdfs:range
rdf:resource="&foaf;Person" /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&foaf;mbox">
<rdfs:domain rdf:resource="&foaf;Person" /> <rdfs:range
rdf:resource="&xsd:string" /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="worksWith"> <rdf:type
rdf:resource="&owl;SymmetricProperty" /> <rdfs:subPropertyOf
rdf:resource="&foaf;knows" /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="playsTennisWith"> <rdf:type
rdf:resource="&owl;SymmetricProperty" /> <rdfs:subPropertyOf
rdf:resource="#playsSportWith" /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="playsSportWith"> <rdf:type
rdf:resource="&owl;SymmetricProperty" /> <rdfs:subPropertyOf
rdf:resource="&foaf;knows" /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="marriedWith"> <rdf:type
rdf:resource="&owl;SymmetricProperty" /> <rdf:type
rdf:resource="&owl;FunctionalProperty" /> <rdf:type
rdf:resource="&owl;InverseFunctionalProperty" />
<rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>
```

We would like to express that an email address cannot correspond to more than one person by the property *foaf:mbox*.

2. How to express this in OWL: with a cardinality constraint, a functional property, an inverse functional property or a symmetric property?

This is an inverse functional property which states that to an email address corresponds only one image by the inverse of *foaf:mbox*. This is simply expressed by¹:

```
<owl:InverseFunctionalProperty rdf:about="&foaf;mbox" />
```

It is possible to use:

```
<owl:Class rdf:about="&owl;Thing">
<rdfs:subClassOf> <owl:Restriction> <owl:onProperty>
<owl:ObjectProperty> <owl:inverseOf rdf:resource="&foaf;mbox"
/> </owl:ObjectProperty> </owl:onProperty>
<owl:maxCardinality>1</owl:maxCardinality> </owl:Restriction>
</rdfs:subClassOf> </owl:Class>
```

¹In principle, in OWL 1, this only works on ObjectProperties and *foaf:mbox* is a DataProperty.

3. Is it possible to express this constraint in DL-Lite? If yes, how?

This is possible in the DL-Lite_F language:

```
(functional foaf:mbox-1)
```

4. Consider the graphs of Figure 2.6 (p21) to which this constraint is added, does this change something to the answers given to Question 3 of Exercise 2? (explain why)

(a) becomes a super-graph of (b), because ?b and ?d becomes the same node because they have the foaf:mbox "keith@g.es" in common and this an inverse functional property. The only difference between both graphs is that ⟨?a, foaf:knows, ?b⟩ belongs to (a) and not to (b). As a consequence, $a, O' \models b$ (with O' the ontology with the inverse functional property constraint), but not the other way around for the same reason as before.

Exercise 7 (RDFS ontologies). Consider the RDFS ontology o containing, in addition to those of the graph G of Exercise 3, the following statements:

```
⟨o:Novel, rdfs:subClassOf, o:Literature⟩
⟨o:Poem, rdfs:subClassOf, o:Literature⟩
⟨o:translated, rdfs:range, o:Literature⟩
⟨o:wrote, rdfs:domain, o:Writer⟩
```

1. Does this allow to conclude that $d:Poe$, $d:Baudelaire$ or $d:Mallarmé$ is a $o:Writer$? Explain why.

The only assertion that would allow to conclude that someone is a $o:Writer$ is the last one related to the domain of the $o:wrote$ predicate. Nothing allows for inferring triples with the $o:wrote$ predicate, so the only assertions with it are those asserted. Hence, the only writers are $d:Poe$ and $d:Mallarmé$.

2. Can you express in OWL the statement that “anyone who write Literature is a Writer”?

The sentence expresses that those who write Literature are Writers, hence Writer is a superclass of the restriction. This can be expressed by creating a class equivalent to the restriction, and subclass of Writer:

```
<owl:Class> <owl:equivalentClass> <owl:Restriction>
<owl:onProperty rdf:resource="o:wrote"/> <owl:someValueFrom
rdf:resource="o:Literature"/> </owl:Restriction>
</owl:equivalentClass> <rdfs:subClassOf rdf:resource="o:Writer"/>
</owl:Class>
```

or more briefly:

```
<owl:Restriction> <owl:onProperty rdf:resource="o:wrote"/>
<owl:someValueFrom rdf:resource="o:Literature"/>
<rdfs:subClassOf rdf:resource="o:Writer"/> </owl:Restriction>
```

It would have been possible to add a range constraint on the $o:wrote$ predicate so that whatever is written is $o:Literature$ (⟨ $o:wrote$, rdfs:range, $o:Literature$ ⟩). However, this is stronger than what was asked since it would have restricted a particular author to write something else than literature.

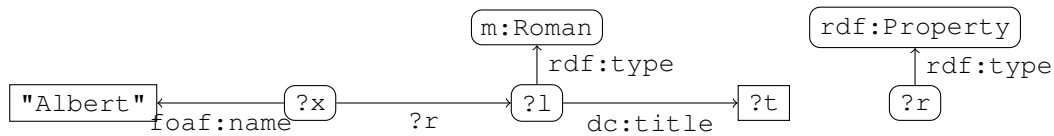


Figure 14.3: SPARQL graph patterns.

SPARQL

Exercise 8 (SPARQL queries). *Given the following SPARQL query:*

```
SELECT ?t
PREFIX
  foaf: http://xmlns.com/foaf/0.1/
  m: http://mydomain.com/myExample#
  dc: http://purl.org/dc/elements/1.1/
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
WHERE
  ?x foaf:name "Albert".
  ?x ?r ?l.
  ?r rdf:type rdf:Property.
  ?l rdf:type m:Roman.
  ?l dc:title ?t.
```

1. *What is the informal meaning of this query?*

Find all titles (`dc:title`) of novels (`m:Roman`) related (`?r`) to an entity (`?x`) whose name (`foaf:name`) is “Albert”.

2. *Draw the RDF graph corresponding to the graph patterns of the query.*

It is given in Figure ??.

3. *What is the difference between such graph patterns and simple RDF graphs?*

These graph patterns are generalised RDF graphs: they can contain blanks or variables as edge labels (or as properties in RDF triple terms).

4. *Evaluate this query on each of the well-founded graphs of Figure 2.5 of Exercise 1 (p20) and provide the answer.*

(a) : { { "La peste" } }

(c) : { } (`?l` is a book (`Livre`) but not a novel (`Roman`))

(d) : { { "La peste" } }

5. *What must be added to this query for returning the year of publication of the `m:Roman` if it is available?*

An optional clause:

```
SELECT ?t ?y ...OPTIONAL { ?l m:annee ?y. }
```

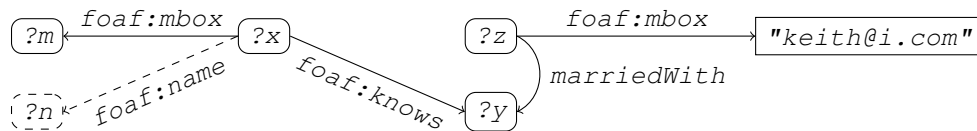
Exercise 9 (SPARQL and queries). *Given the following SPARQL query:*

```

SELECT ?m, ?n
PREFIX foaf: http://xmlns.com/foaf/0.1/
WHERE {
  ?x foaf:mbox ?m.
  ?x foaf:knows ?y.
  ?z foaf:mbox "keith@i.com".
  ?z marriedWith ?y.
  OPTIONAL { ?x foaf:name ?n }
}

```

1. What is, informally, the meaning of this query (tell it in English or French)?
What are the email address, and if available the name, of people who know the spouse of the person with email address "keith@i.com"?
2. Draw the GRDF graph corresponding to the graph pattern of this query.



3. Evaluate the query on the graphs of Figure 2.6 of Exercise 2 (p21) and provide the results.
For both graphs, there is no result because there is no node which is both the object of a `foaf:knows` and a `marriedWith` (for matching `?y`).
4. Evaluate this query on the closure obtained at Question 4 of Exercise 5 and provide the results.
(a) still no result because this time there is no node whose `foaf:mbox` is "keith@i.com" which is the subject of a `marriedWith` relation, (b) 1 result: $\{ \langle ?m, \text{"paul@e.at"} \rangle, \langle ?n, \text{null} \rangle \}$ because, $\langle ?a, \text{foaf:knows}, ?c \rangle$ belongs to the closure of (b).

Exercise 10 (SPARQL query containment). Consider the following queries q_1 and q_2 on the RDF graph G of Exercise 3:

$$q_1 = \text{SELECT } ?w \text{ FROM } G \text{ WHERE } (\langle ?w \text{ o:wrote } ?x \rangle \text{ AND } \langle ?x \text{ rdf:type o:Poem} \rangle) \cup \langle ?w \text{ o:translated } ?x \rangle$$

$$q_2 = \text{SELECT } ?w \text{ FROM } G \text{ WHERE } (\langle ?w \text{ o:wrote } ?l \rangle \cup \langle ?w \text{ o:translated } ?l \rangle) \text{ AND } \langle ?l \text{ rdf:type o:Poem} \rangle$$

1. In the course, we defined the distinguished variables \vec{B} , the queried graph G and the query pattern P . Identify them in q_1 and q_2 .
In both cases, \vec{B} is $\langle ?w \rangle$ and G is G . The patterns of q_1 and q_2 are respectively :
 $P_1 = (\langle ?w \text{ o:wrote } ?x \rangle \text{ AND } \langle ?x \text{ rdf:type o:Poem} \rangle) \cup \langle ?w \text{ o:translated } ?x \rangle$
 $P_2 = (\langle ?w \text{ o:wrote } ?l \rangle \cup \langle ?w \text{ o:translated } ?l \rangle) \text{ AND } \langle ?l \text{ rdf:type o:Poem} \rangle$

2. Provide the answers of q_1 and q_2 with respect to the graph G .

The answers to query q_1 and q_2 on the graph G are respectively $\{\langle d:Poe \rangle, \langle d:Mallarmé \rangle, \langle d:Baudelaire \rangle\}$ and $\{\langle d:Poe \rangle, \langle d:Mallarmé \rangle\}$.

Query containment $q \sqsubseteq q'$ between two queries $q = \text{SELECT } \vec{B} \text{ FROM } G \text{ WHERE } P$ and $q' = \text{SELECT } \vec{B}' \text{ FROM } G \text{ WHERE } P'$ is defined by the fact that for any RDF graph, the answers to q are included in those to q' ($\forall G, \mathcal{A}(\vec{B}, G, P) \subseteq \mathcal{A}(\vec{B}', G, P')$).

3. What does the answer to the previous questions tell you about query containment between q_1 and q_2 ?

$q_1 \not\sqsubseteq q_2$ because G is a counter example:
 $\{\langle d:Poe \rangle, \langle d:Mallarmé \rangle, \langle d:Baudelaire \rangle\} = \mathcal{A}(\langle ?w \rangle, G, P_1) \not\subseteq \mathcal{A}(\langle ?w \rangle, G, P_2) = \{\langle d:Poe \rangle, \langle d:Mallarmé \rangle\}$.

4. Do you think that query containment holds in some direction between q_1 and q_2 (either $q_1 \sqsubseteq q_2$ or $q_2 \sqsubseteq q_1$)?

$q_2 \sqsubseteq q_1$.

5. Provide a proof for this. This may be done semantically by using the interpretation of query patterns or syntactically by translating queries into logic and showing that the query containment statement is a theorem.

The argument for the proof is that $P_1 = (A \wedge B) \vee C$ and $P_2 = (A \vee C) \wedge B$, but $(A \vee C) \wedge B = (A \wedge B) \vee (C \wedge B)$. Hence, P_2 is more specific than P_1 . More formally:

$$\begin{aligned} & \sigma \in \mathcal{A}(\langle ?w \rangle, G, P_2) \\ \Leftrightarrow & G \models \sigma((\langle ?w \text{ o: } \text{wrote } ?l \rangle \text{ UNION } \langle ?w \text{ o: } \text{translated } ?l \rangle) \text{ AND } \langle ?l \text{ rdf:type o:Poem} \rangle) \\ \Leftrightarrow & G \models \sigma(\langle ?w \text{ o: } \text{wrote } ?l \rangle \text{ UNION } \langle ?w \text{ o: } \text{translated } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \text{ rdf:type o:Poem} \rangle) \\ \Leftrightarrow & (G \models \sigma(\langle ?w \text{ o: } \text{wrote } ?l \rangle) \text{ or } G \models \sigma(\langle ?w \text{ o: } \text{translated } ?l \rangle)) \text{ and } G \models \sigma(\langle ?l \text{ rdf:type o:Poem} \rangle) \\ \Leftrightarrow & (G \models \sigma(\langle ?w \text{ o: } \text{wrote } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \text{ rdf:type o:Poem} \rangle)) \\ & \text{ or } (G \models \sigma(\langle ?w \text{ o: } \text{translated } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \text{ rdf:type o:Poem} \rangle)) \\ \Rightarrow & (G \models \sigma(\langle ?w \text{ o: } \text{wrote } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \text{ rdf:type o:Poem} \rangle)) \text{ or } G \models \sigma(\langle ?w \text{ o: } \text{translated } ?l \rangle) \\ \Leftrightarrow & G \models \sigma(\langle ?w \text{ o: } \text{wrote } ?x \rangle \text{ AND } \langle ?x \text{ rdf:type o:Poem} \rangle) \text{ or } G \models \sigma(\langle ?w \text{ o: } \text{translated } ?x \rangle) \\ \Leftrightarrow & G \models \sigma((\langle ?w \text{ o: } \text{wrote } ?x \rangle \text{ AND } \langle ?x \text{ rdf:type o:Poem} \rangle) \text{ UNION } \langle ?w \text{ o: } \text{translated } ?x \rangle) \\ \Leftrightarrow & \sigma \in \mathcal{A}(\langle ?w \rangle, G, P_1) \end{aligned}$$

Hence, $q_2 \sqsubseteq q_1$.

Query modulo ontologies

Exercise 11 (SPARQL modulo ontologies). 1. The way queries are answered in Exercise 9 is not the standard way for answering queries modulo DL-Lite ontologies. What is the standard method? Rewrite the above query with this method and give its results.

The usual way to evaluate queries modulo DL-Lite is to rewrite the query as the following:

```

SELECT ?m, ?n PREFIX foaf: http://xmlns.com/foaf/0.1/
WHERE { ?x foaf:mbox ?m. { { ?x foaf:knows ?y. } UNION { ?x
foaf:playsSportWith ?y. } UNION { ?x foaf:playsTennisWith ?y.
} UNION { ?x foaf:worksWith ?y. } UNION { ?x marriedWith ?y.}
} ?z foaf:mbox "keith@i.com". ?z marriedWith ?y. OPTIONAL { ?x
foaf:name ?n } }

```

It can be checked that this query will return the same results as in Question 4 of Exercise 9.

Exercise 12 (Query modulo ontology). We now consider the ontology o of Exercise 7 and the following queries:

- $q_3 = \text{SELECT } ?y \text{ FROM } o \text{ WHERE } \langle ?x, o:\text{translated}, ?y \rangle$;
- $q_4 = \text{SELECT } ?y \text{ FROM } o \text{ WHERE } \langle ?y, \text{rdf:type}, o:\text{Literature} \rangle$.

1. Do you think that query containment holds in some direction between q_3 and q_4 (either $q_3 \sqsubseteq q_4$ or $q_4 \sqsubseteq q_3$)? Tell why.

None of these because SPARQL evaluates queries by finding triples in the graph and the triples are not comparable. More formally, assume $G_1 = \{\langle a, o:\text{translated}, b \rangle\}$ and $G_2 = \{\langle c, \text{rdf:type}, o:\text{Literature} \rangle\}$, it is clear that $\mathcal{A}(q_3, G_1) \not\subseteq \mathcal{A}(q_4, G_1)$ and $\mathcal{A}(q_4, G_2) \not\subseteq \mathcal{A}(q_3, G_2)$. Hence, there cannot be any containment between these queries.

2. Can you provide a definition for query containment modulo an ontology o ($q \sqsubseteq_o q'$)?

There is no reason to change the structure of the definition: Query containment $q \sqsubseteq q'$ between two queries $q = \text{SELECT } \vec{B} \text{ FROM } o \text{ WHERE } P$ and $q' = \text{SELECT } \vec{B}' \text{ FROM } o \text{ WHERE } P'$ is defined by the fact that for any RDFS ontologies, the answers to q are included in those to q' ($\forall o, \mathcal{A}^+(\vec{B}, o, P) \subseteq \mathcal{A}^+(\vec{B}', o, P')$).

Everything is in the definition of \mathcal{A}^+ . A natural semantic definition would be that:

$$\mathcal{A}^+(\vec{B}, o, P) = \{\sigma|_{\vec{B}} \mid o \models_{RDFS} \sigma(P)\}$$

or a more pragmatic approach would be to define it with the closure:

$$\mathcal{A}^+(\vec{B}, o, P) = \mathcal{A}(\vec{B}, \hat{o} \setminus P, P)$$

3. Does it return different answers for q_3 and q_4 ? (either $q_3 \sqsubseteq_o q_4$ or $q_4 \sqsubseteq_o q_3$)? Tell why.

From the definition of o , it is clear that whenever $o \models_{RDFS} \langle a, o:\text{translated}, b \rangle$, then $o \models_{RDFS} \langle b, \text{rdf:type}, o:\text{Literature} \rangle$. The converse is not true (there exists models satisfying $\langle b, \text{rdf:type}, o:\text{Literature} \rangle$ but no a such that $\langle a, o:\text{translated}, b \rangle$, i.e., there may be non translated books). Then for any $\langle b \rangle \in \mathcal{A}^+(q_3)$, $\langle b \rangle \in \mathcal{A}^+(q_4)$, so $q_3 \sqsubseteq_o q_4$. But not the other way around.

Network of ontologies

Exercise 13 (Network of ontologies). In addition of the ontology o of Exercise 7, we consider an ontology o' which defines the class $op:\text{Buch}$ and contains the following statements:

$\langle d:\text{Baudelaire}, o:\text{translated}, d:\text{Confessions} \rangle \langle d:\text{DeQuincey}, o:\text{wrote}, d:\text{Confessions} \rangle$

and o'' which defines the class $opp:Roman$ and contain the following statements:

$\langle d:Confessions, rdf:type, opp:Roman \rangle \langle d:Musset, o:translated, d:Confessions \rangle$

They are related together by the following three alignments:

- $A_{o,o'} = \{ \langle o:Literature, \equiv, op:Buch \rangle \}$
- $A_{o',o''} = \{ \langle op:Buch, \sqsubseteq, opp:Roman \rangle \}$
- $A_{o'',o} = \{ \langle opp:Roman, \equiv, o:Novel \rangle \}$

So that we have a network of ontology $\langle \{o, o', o''\}, \{A_{o,o'}, A_{o',o''}, A_{o'',o}\} \rangle$.

1. Do you think that this network of ontologies is well designed? Why?

It is correctly defined because it is made a set of ontologies and a set of alignments between these ontologies. However, the statement $op:Buch \sqsubseteq opp:Roman$ seems strange and maybe exactly the opposite.

2. Is this network consistent? Provide a model for this network of ontologies.

The network is indeed consistent. As a model it is possible to create a model isomorphic to the ontologies (with, in each of the ontologies, the same URI interpreted in the same way and equivalent classes having the same interpretation). A model of the network may have been a triple $\langle m, m', m'' \rangle$ such that:

$$\begin{aligned}
 m(o:Literature) &= m'(op:Buch) = m''(opp:Roman) = m(o:Novel) \\
 m(op:Poem) &\subseteq m(o:Literature) \\
 \{m(d:Poe), m(d:Mallarmé), m'(d:DeQuincey)\} &\subseteq m(Writer) \\
 \{m''(d:Confessions), m(d:TheGoldBug)\} &\subseteq m(o:Literature) \\
 \{m(d:TheRaven), m(d:Brise marine)\} &\subseteq m(o:Poem) \\
 m(.:b) &= m(d:Brise marine) \\
 \langle m(d:Poe), m(d:TheGoldBug) \rangle &\in m(o:wrote) \\
 \langle m(d:Poe), m(d:TheRaven) \rangle &\in m(o:wrote) \\
 \langle m(d:Mallarmé), m(d:Brise Marine) \rangle &\in m(o:wrote) \\
 \langle m'(d:DeQuincey), m'(d:Confessions) \rangle &\in m'(o:wrote) \\
 \langle m(d:Mallarmé), m(d:TheRaven) \rangle &\in m(o:translated) \\
 \langle m(d:Baudelaire), m(TheGoldBug) \rangle &\in m(o:translated) \\
 \langle m''(d:Musset), m''(d:Confessions) \rangle &\in m''(o:translated) \\
 \langle m'(d:Baudelaire), m'(d:Confessions) \rangle &\in m'(o:translated)
 \end{aligned}$$

3. Provide the constraints that the alignments impose on models.

The constraints are that:

$$\begin{aligned}
 m(o:Literature) &= m'(op:Buch) \\
 m'(op:Buch) &\subseteq m''(opp:Roman) \\
 m''(opp:Roman) &= m(o:Novel)
 \end{aligned}$$

but since $m(o:Novel) \subseteq m(o:Literature)$, we have $m(o:Literature) = m'(op:Buch) = m''(opp:Roman) = m(o:Novel)$.

-
4. What does this entail for the class (*rdf:type*) of *d:Confessions* and *d:TheRaven* at *o* in this network?

This entails that both works have all these four classes as rdf:type. In particular, $\langle d:TheRaven, rdf:type, o:Poem \rangle$ and $\langle d:TheRaven, rdf:type, o:Novel \rangle$.

Semantic peer-to-peer systems

Exercise 14 (Semantic peer-to-peer system). *Given a peer-to-peer system with peers n_1, n_2 and n_3 all related to each others². Peers n_1 and n_2 share the ontology *o* that has been defined at Question 1 of Exercise 4, n_3 uses the ontology *o'* defining;*

```
t:Work rdf:type owl:Class.
t:copyrightHolder rdf:type rdf:Property.
t:copyrightHolder rdfs:domain t:Work.
t:year rdf:type rdf:Property.
t:year rdfs:domain t:Work.
t:year rdfs:range xsd:integer.
```

*There exists an alignment *A* between *o* and *o'* containing two correspondences: $t:Work \geq m:Livre$, $t:year \equiv m:annee$. Assume that n_1 does not contain instances; n_2 contains:*

```
http://mm.com#a345 m:aecrit http://isbn.org/2070360423.
http://mm.com#a345 m:aecrit http://isbn.org/2070360024.
http://mm.com#a345 m:aecrit http://isbn.org/2070322882.
http://mm.com#a345 foaf:name "Albert".
http://isbn.org/2070360423 rdf:type m:Roman.
http://isbn.org/2070360423 dc:title "La peste".
http://isbn.org/2070360024 rdf:type m:Roman.
http://isbn.org/2070360024 dc:title "L' étranger".
http://isbn.org/2070322882 rdf:type m:Livre.
http://isbn.org/2070322882 dc:title "Le Mythe de Sisyphe".
...
```

and n_3 contains:

```
http://isbn.org/2070360423 t:year 1947.
http://isbn.org/2070322882 t:year 1942.
...
```

1. Express the alignment *A* in OWL.

```
<owl:Class rdf:about="m:Livre"> <rdfs:subClassOf
rdf:resource="t:Work"/> </owl:Class>
<owl:DataProperty rdf:about="t:year"> <owl:equivalentProperty
rdf:resource="m:annee"/> </owl:DataProperty>
```

2. The peer n_1 would like to evaluate the following query:

```
SELECT ?t, ?y
PREFIX ...
WHERE
```

²This exercise has not been given at the actual exam.

```
?x foaf:name "Albert".
?x m:aecrit ?l.
?l rdf:type m:Roman.
?l dc:title ?t.
OPTIONAL { ?l m:annee ?y. }
```

How is it possible to answer this query by using n_2 , n_3 and A ?

It can be evaluated directly in n_1 and n_2 . It can be translated with the help of A for being evaluated in n_3 . However, the three data base do not contain the same type of information: n_1 contains no data, n_2 contains information about authors and titles while n_3 contains information about publication year. Hence, instead of transforming the query for evaluating it on n_3 , it is better to split it into two queries.

The result of the transformation would be:

```
SELECT ?l, ?t PREFIX ... WHERE ?x foaf:name "Albert". ?x
m:aecrit ?l. ?l rdf:type m:Roman. ?l dc:title ?t.
```

and

```
SELECT ?l, ?y PREFIX ... WHERE OPTIONAL { ?l m:annee ?y }
```

The second query can be transformed thanks to A into:

```
SELECT ?l, ?y PREFIX ... WHERE OPTIONAL { ?l t:year ?y }
```

and their results can be joined. This is not particularly efficient but this should work. It is also possible to first evaluate the former query to retrieve $?l$ and then to issue queries for specific values of the first answer.

3. Provide the answer on the available data.

Evaluating directly the query on n_2 would yield:

```
{<"La peste",null>,
 <"L'étranger",null>,...}
```

while evaluating a transformation of this query against n_3 would yield an empty answer set.

Using the strategy provided above would return the results:

```
{<"La peste",1947>,
 <"L'étranger",null>,...}
```

Bibliography

- [Abiteboul *et al.*, 2011] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Web data management*. Cambridge university press, Cambridge (UK), 2011. 39
- [Alkhateeb *et al.*, 2009] Faisal Alkhateeb, Jean-François Baget, and Jérôme Euzenat. Extending SPARQL with regular expression patterns (for querying RDF). *Journal of web semantics*, 7(2):57–73, 2009. 18, 55, 61, 65
- [Alkhateeb, 2008] Faisal Alkhateeb. *Querying RDF(S) with regular expressions*. Thèse d’informatique, Université Joseph Fourier, Grenoble (FR), 2008. 6, 18, 19, 70
- [Allen, 1983] James Allen. Maintaining knowledge about temporal intervals. *Communication of the ACM*, 26(11):832–843, 1983. 88
- [Antoniou and van Harmelen, 2008] Grigoris Antoniou and Frank van Harmelen. *A semantic web primer*. The MIT press, 2 edition, 2008. 6
- [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, Cambridge (UK), 2003. 31
- [Baget, 2003] Jean-François Baget. Homomorphismes d’hypergraphes pour la subsumption en RDF. In *Proc. 3e journées nationales sur les modèles de raisonnement (JNMR), Paris (France)*, pages 1–24, 2003. 30
- [Baget, 2005] Jean-François Baget. RDF entailment as a graph homomorphism. In *Proceedings of the 4th International Semantic Web Conference (ISWC’05), Galway (IE)*, pages 82–96, 2005. 11, 14, 17, 18, 20
- [Bao *et al.*, 2009] Jie Bao, Elisa Kendall, Deborah McGuinness, and Peter Patel-Schneider. OWL 2 web ontology language: quick reference guide. Recommendation, W3C, 2009. <http://www.w3.org/TR/owl2-quick-reference/>. 31
- [Beckett, 2006] Dave Beckett. Turtle - terse RDF triple language. Technical report, Hewlett-Packard, Bristol (UK), 2006. 12
- [Beckett, 2009a] Dave Beckett. OWL 2 web ontology language document overview. Recommendation, W3C, October 2009. 11, 31
- [Beckett, 2009b] Dave Beckett. RDF/XML syntax specification (revised). Recommendation, W3C, February 2009. 12

- [Berners-Lee *et al.*, 1998] Tim Berners-Lee, Roy Fielding, and L. Masinter. Uniform resource identifiers (URI): Generic syntax. RFC 2396, IETF, 1998. <http://www.ietf.org/rfc/rfc2396.txt>. 11
- [Borgida and Serafini, 2003] Alexander Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *Journal on Data Semantics*, I:153–184, 2003. 98, 105
- [Bouquet *et al.*, 2004] 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. 79
- [Brickley and Guha, 2004] Dan Brickley and Ramanathan Guha. RDF vocabulary description language 1.0: RDF schema. Recommendation, W3C, 2004. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. 11, 14, 23
- [Broekstra, 2003] Jeen Broekstra. SeRQL: Sesame RDF query language. In *SWAP Deliverable 3.2 Method Design*, 2003. 51
- [Calvanese *et al.*, 2000] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Vardi. View-based query processing for regular path queries with inverse. In *Proceedings of the 19th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 58–66, 2000. 70
- [Calvanese *et al.*, 2002] 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. 105
- [Calvanese *et al.*, 2004] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Logical foundations of peer-to-peer data integration. In *Proc. 23rd Symposium on Principles of Database Systems (PODS)*, pages 241–251, Paris (FR), 2004. 105
- [Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: the DL-Lite family. *Journal of automated reasoning*, 39(3):385–429, 2007. 38, 41, 71
- [Carroll and Klyne, 2004] Jeremy Carroll and Graham Klyne. RDF concepts and abstract syntax. Recommendation, W3C, February 2004. 11, 12
- [Chein and Mugnier, 2009] Michel Chein and Marie-Laure Mugnier. *Graph-based knowledge representation*. Springer, Heidelberg (DE), 2009. 18
- [Corby *et al.*, 2000] Olivier Corby, Rose Dieng, and Cédric Hébert. A conceptual graph model for W3C resource description framework. In *Proceedings of the International Conference on Conceptual Structures*, pages 468–482, 2000. 18
- [Cudré-Mauroux, 2008] Philippe Cudré-Mauroux. *Emergent Semantics: Interoperability in large-scale decentralized information systems*. EPFL Press, Lausanne (CH), 2008. 109

- [De Leenheer and Mens, 2008] Pieter De Leenheer and Tom Mens. Ontology evolution; state of the art and future directions. 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. 76
- [Dean and Schreiber, 2004] Mike Dean and Guus Schreiber. OWL web ontology language: reference. Recommendation, W3C, February 2004. 31
- [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. 93
- [Euzenat *et al.*, 2007] Jérôme Euzenat, François Scharffe, and Antoine Zimmermann. Expressive alignment language and implementation. Deliverable D2.2.10, Knowledge Web Network of Excellence, 2007. 76, 77, 79, 81
- [Euzenat, 2001] Jérôme Euzenat. Granularity in relational formalisms with application to time and space representation. *Computational intelligence*, 17(4):703–737, 2001. 94
- [Euzenat, 2002] Jérôme Euzenat. Sémantique des représentations de connaissance. Notes de cours, 2002. 3
- [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. 83
- [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. 79
- [Euzenat, 2007] Jérôme Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 348–353, Hyderabad (IN), 2007. 92, 93, 104
- [Euzenat, 2008] Jérôme Euzenat. Algebras of ontology alignment relations. In *Proc. 7th conference on international semantic web conference (ISWC), Karlsruhe (DE)*, pages 387–402, 2008. 80, 88
- [Fagin *et al.*, 1995] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. The MIT press, Cambridge (MA US), 1995. 102, 103, 105
- [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. 83
- [Freksa, 1992] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial intelligence*, 54(1):199–227, 1992. 94
- [Gal *et al.*, 2005] Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, and Danilo Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 14(1):50–67, 2005. 81

- [Ghidini and Giunchiglia, 2001] Chiara Ghidini and Fausto Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001. 103
- [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. 83, 98, 105, 106
- [Glimm and Ogbuji, 2010] Birte Glimm and Chimezie Ogbuji. SPARQL 1.1 entailment regimes. Working draft, W3C, June 2010. <http://www.w3.org/TR/sparql11-entailment>. 57, 65
- [Gottlob *et al.*, 1999] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural CSP decomposition methods. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 394–399, 1999. 20
- [Grahne and Thomo, 2003] Gösta Grahne and Alex Thomo. Query containment and rewriting using views for regular path queries under constraints. In *Proc. 22nd ACM symposium on Principles of database systems (PODS)*, pages 111–122, New-York (NY US), 2003. ACM. 70
- [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. 98
- [Gutierrez *et al.*, 2004] Claudio Gutierrez, Carlos Hurtado, and Alberto Mendelzon. Foundations of semantic web databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 95–106, 2004. 11, 18, 19, 30, 56
- [Hayes, 2004] Patrick Hayes. RDF semantics. Recommendation, W3C, February 2004. 11, 12, 14, 16, 17, 18, 23, 25, 28, 29, 30
- [Hitzler *et al.*, 2009] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman & Hall/CRC., 2009. 6
- [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. 38
- [Karvounarakis *et al.*, 2002] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. RQL: A declarative query language for RDF. In *Proceedings of the 11th International Conference on the World Wide Web (WWW2002)*, 2002. 51
- [Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. 21st Symposium on Principles of Database Systems (PODS)*, pages 233–246, Madison (WI US), 2002. 98
- [Ligozat and Renz, 2004] Gérard Ligozat and Jochen Renz. What is a qualitative calculus? a general framework. In *Proc. 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI), Auckland (NZ)*, number 3157 in Lecture notes in computer science, pages 53–64, 2004. 88

- [Manola and Miller, 2004] Frank Manola and Eric Miller. RDF primer. Recommendation, W3C, 2004. <http://www.w3.org/TR/REC-rdf-syntax/>. 11
- [Motik *et al.*, 2009a] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 web ontology language: profiles. Recommendation, W3C, 2009. <http://www.w3.org/TR/owl2-profiles/>. 37, 41
- [Motik *et al.*, 2009b] Boris Motik, Peter Patel-Schneider, and Bernardo Cuenca Grau. OWL 2 web ontology language: direct semantic. Recommendation, W3C, 2009. <http://www.w3.org/TR/owl2-direct-semantics/>. 33
- [Muñoz *et al.*, 2007] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. Minimal deductive systems for RDF. In *Proceedings of the 4th European Semantic Web Conference(ESWC), Innsbruck (AT)*, pages 53–67, 2007. <http://www.springerlink.com/content/g8w64n1264874118/>. 65, 66, 67
- [Muñoz *et al.*, 2009] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. Simple and efficient minimal RDFS. *Journal of web semantics*, 7(3):220–234, 2009. 65
- [Papakonstantinou and Vassalos, 1999] Yannis Papakonstantinou and Vasilis Vassalos. Query rewriting for semistructured data. In *Proc. ACM international conference on Management of data (SIGMOD), Philadelphia (PA US)*, pages 455–466, ACM Press, New-York (NY US), 1999. 70
- [Patel-Schneider *et al.*, 2004] Peter Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. Recommendation, W3C, February 10 2004. 33
- [Pérez *et al.*, 2006] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. In *Proceedings of the 5th International Semantic Web Conference*, pages 30–43, Athens (GA US), 2006. 18, 19, 51, 53, 54, 56
- [Pérez *et al.*, 2008] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. nSPARQL: A navigational language for RDF. In *Proceedings of the 7th International Semantic Web Conference, Karlsruhe (DE)*, pages 66–81, 2008. 67, 68, 69
- [Pérez *et al.*, 2009] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM transactions on database systems*, 34(3):16, 2009. 56
- [Prud’hommeaux and Seaborne, 2008] Eric Prud’hommeaux and Andy Seaborne. SPARQL query language for RDF. Recommendation, W3C, January 2008. 51, 53, 79
- [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. 82
- [Rousset *et al.*, 2006] Marie-Christine Rousset, Philippe Adjiman, Philippe Chatalic, François Goasdoué, and Laurent Simon. Somewhere in the semantic web. In *Proc. 32nd International Conference on Current Trends in Theory and Practice of Computer Science (SofSem)*, volume 3831 of *Lecture notes in computer science*, pages 84–99, Merin (CZ), 2006. 76
- [Seaborne, 2004] Andy Seaborne. RDQL - a query language for RDF. Member submission, W3C, 2004. 51

- [Sheth and Larson, 1990] Amit Sheth and James Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990. 82
- [Tarski, 1941] Alfred Tarski. On the calculus of relations. *Journal of symbolic logic*, 6(3):73–89, 1941. 88
- [ter Horst, 2004] Herman ter Horst. Extending the RDFS entailment lemma. In *Proceedings of the 3rd International Semantic web Conference (ISWC)*, pages 77–91, 2004. 20
- [ter Horst, 2005] Herman ter Horst. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2):79–115, 2005. 13, 14, 17, 30, 66
- [Wooldridge, 2000] Mike Wooldridge. *Reasoning about rational agents*. The MIT press, Cambridge (MA US), 2000. 105
- [Zimmermann and Euzenat, 2006] Antoine Zimmermann and Jérôme Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *Proc. 5th conference on International semantic web conference (ISWC), Athens (GA US)*, pages 16–29, 2006. 98, 104, 105
- [Zimmermann and Le Duc, 2008] Antoine Zimmermann and Chan Le Duc. Reasoning on a network of aligned ontologies. In *Proc. 2nd International conference on web reasoning and rule systems (RR), Karlsruhe (DE)*, pages 43–57, 2008. 92
- [Zimmermann et al., 2006] Antoine Zimmermann, Markus Krötzsch, Jérôme Euzenat, and Pascal Hitzler. Formalizing ontology alignment and its operations with category theory. In *Proc. 4th International Conference on Formal Ontology in Information Systems (FOIS)*, pages 277–288, Baltimore (MD US), 2006. 91, 92
- [Zimmermann, 2008] Antoine Zimmermann. *Sémantique des réseaux de connaissances: gestion de l'hétérogénéité fondée sur le principe de médiation*. Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 2008. 6, 98, 104