Classification of concepts through products of concepts and abstract data types

Petko Valtchev, Jérôme Euzenat
INRIA Rhône-Alpes – IMAG-LIFIA
46, avenue Félix Viallet, 38031 Grenoble Cedex 1, France
{Petko.Valtchev,Jerome.Euzenat}@imag.fr

Abstract

A dissimilarity measure is presented which only takes into account a partial order defined on the considered domain. It compares a couple of individuals with respect to their mutual position within the taxonomy structuring that domain. This applies to both usual data types and structured objects. As a consequence it is possible to design a classification algorithm to work on objects whose field values are such data and objects.

1 Introduction

Object-based systems offer the opportunity of representing and organising knowledge into taxonomies of classes. Therefore the application of some data analysis techniques, and precisely the automatic classification algorithms, may be useful for such systems. However, classification algorithms must be able of:

1. coping with various data types,
2. considering objects as a whole (instead of a set of characteristics),
3. respecting the structure of the domains (Diday, 1993), and,
4. providing intelligible class descriptions (Michalski and Stepp, 1983).

Here we tackle the three former problems. First, an extensible type system is obtained through abstract data types. Types and objects are modelled

as classification schemes whose aim is to focus on the taxonomies structuring a domain. Classification schemes allow the design of algorithm able to deal with both (extensible) types and objects. They are provided with a dissimilarity measure exclusively depending on the partial order structure. Thus, when clustering a set of objects (1) nothing has to be known about the domains (types or objects) for computing dissimilarity, (2) the objects are considered as class members rather than structures, and (3) the dissimilarity completely accounts for the class taxonomy it is immediatly computed on.
We present here a way to perform a classification of a set of objects on referring to taxonomies of other objects and hierarchies of externally defined types. The TROPES system and its way to handle external types is first presented (§2). Then, the classification scheme formalism which serves as support for both types and object is summarised (§3), The topological dissimilarity is presented in §4. Finally, we give a brief insight of a practical classification procedure using concept taxonomies in a bottom-up strategy.

2 Concepts and ADT in TROPES

TROPES is an object-based knowledge representation model (Sherpa project, 1995). The individuals are represented as structured objects and partitioned into a set of *concepts*. A concept describes the fields of the objects it represents. Each field is a map from an object to a "value" whose domain is the object concept and whose co-domain is either another concept or an abstract data type (see below). Thus, an object can be seen as a collection of field values (either objects or values). A class describes a subset of concept objects through refinements of field types (the field types are either classes or type expressions).
In TROPES, primitive values (which can be constructed but whose construction is not accessible to TROPES) are associated with types. Some of them are quite standard (e.g. Integers, Booleans), whereas others can be more elaborate (e.g. Date, Nucleic acid sequence). Each of these types is introduced independently from its implementation by an *abstract data type* (ADT). The ADT for a type T is declared in a generic category such as nominal, discrete or totaly ordered. It is described by an equality predicate between values ($=_T$), a typing predicate ($\in_T$) and, sometimes, an order relationship ($\leq_T$).
The structure of ADT can be compared to that of concepts (see Fig. 1) and what holds for ADT holds for concepts. Thus each concept can be though of as a Cartesian product of ADT and other concepts, more elementary ones. Carrying on with this idea a class may be seen as a product of classes and types while an object becomes a tuple made of objects and values.
This analogy is exploited by the classification formalism in order to deal in

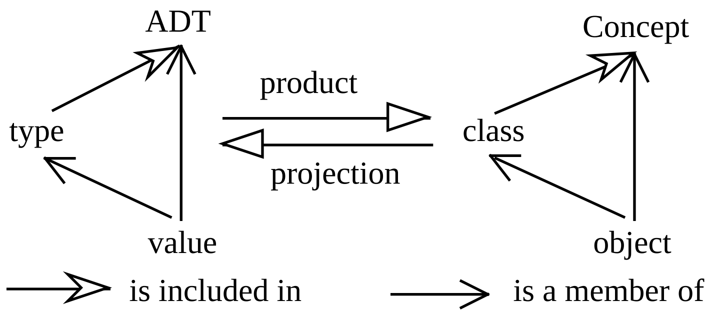

Figure 1: Analogy between concepts and ADT.

a uniform way with ADT and concepts.

3 Classification schemes

A classification scheme is an abstract model of a class taxonomy (Euzenat, 1995). Each classification scheme is defined with regard to a language of individuals L_I and a language of categories L_C. The elements of the first one are interpreted as the entities of a domain to model. The categories have, in turn, two possible interpretations, both as sets of individuals. The abstract interpretation includes all the individuals that the category can represent while the real interpretation contains only those effectively denoted by the category in the modelled domain. In that context, a *taxonomy*, with respect to an interpretation, is defined by a set of categories provided with a partial order. The condition linking the category interpretation to the order relation, called *extension inclusion* property is the following: each time a couple of categories is related by the partial order, set inclusion between the corresponding interpretations holds. A *classification scheme* may be roughly compared to a superposition of two taxonomies. More formally, a classification scheme over a couple of languages L_C and L_I is given by a set of categories ($C \in L_C$) and two partial orders: a sub-categorisation relation ($\leq$) and a sub-categorisation criterion ($\ll$). The sub-categorisation relation constitutes a taxonomy over C with respect to the real interpretation. So does the sub-categorisation criterion over the entire L_C with respect to the abstract interpretation. Furthermore, each time the sub-categorisation relation holds for a couple of categories in C the sub-categorisation criterion must hold too.

Classification schemes can be constructed from more elementary ones through products. A *product* is defined over a set of classification schemes in the fol-

lowing way. Both L_C and L_I of the product are obtained through Cartesian product of the corresponding languages of the basic classification schemes. In a similar way, the partial orders of the product, both the sub-categorisation criterion and the sub-categorisation relation, are defined as order products. A product of classification schemes is a classification scheme (Euzenat, 1995). The reverse operation, called *projection*, singles out one or more classification schemes within a product. This way, the extraction of a field value from a given object is modelled by a unitary projection.
Both operations constitute the theoretical basis for the TROPES bottom-up concept construction (Capponi et al., 1995). ADT in TROPES are elementary classification schemes in which both orders coincide. Concepts are, in turn, products of classification schemes representing ADT or other concepts.
The classification scheme model provides a homogeneous manner to account for the different sub-structures of a concept. This homogeneity has a direct impact on usually performed manipulations over taxonomies like the identification of individuals or the classification of a conceptually described class. For instance, the identification of a constructed object is performed within a product of more elementary classification schemes. The object components are identified under the corresponding classification schemes obtained by projection. The results at the lower levels (corresponding to a concept or an ADT) are used to compose (through product) the global result. Notice that the distinction between concepts, ADT for dense infinite types or ADT for finite ordered types is hidden for the identification algorithm which only manipulates categories and predicates. Here, the same principles of overall homogeneity are applied to automatic classification.

4 Dissimilarity

The complete integration of automatic classification algorithms into the above model requires its extension by a notion of dissimilarity between individuals and/or categories. The category dissimilarity δ is an index measured over the taxonomy ($\delta : C \times C \longrightarrow \mathbb{R}_+$). However, this measure must be based on the sole classification scheme, i.e. on the taxonomy.
The available dissimilarity measures do not fit to our model since they depend strongly on the data types they are applied to (e.g. for heterogeneous data types: Michalski and Stepp, 1983; Esposito, 1994). For instance, the functions used over integer variables are quite different from what is used over strings or Boolean. Furthermore, they distinguish between domains of objects – i.e. concepts – and the ADT (Gowda and Diday, 1994), modelled in a homogeneous manner by the classification schemes. What we need in fact is a measure which holds for any classification scheme regardless of the domain modelled.

The dissimilarity between two entities (either categories or individuals) is the normalised length of the shortest path between them in the taxonomy. The normalisation factor is maximum such length between a couple of individuals. Its exact value is measured over a graph $G = \langle C \cup C_s\ \dot{\leq}\rangle$ made of the taxonomy $\langle C \quad \leq\rangle$ extended by the singleton categories C_s (i.e. the categories whose abstract interpretation is a singleton: the leaves of the tree in Fig. 2) and completed with regard to $\ll$. In the following, singleton categories and individuals are not distinguished.

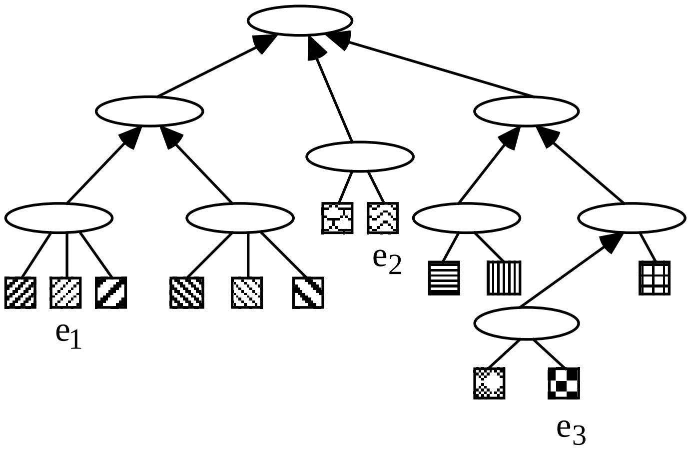

Figure 2: A sample taxonomy.

In the definition of the elementary dissimilarity we shall use the following notations. First, we shall consider a sequence of categories $p = (c_0, c_1 \dots c_m)$ where $c_0 \in C \cup C_s$, $(c_i \in C)_{i=1\dots m}$. Such a sequence is called a *path* if:

$$\forall i, 0 \leq i < m, c_i \dot{<} c_{i+1} \wedge [\forall c \in C, c_i \dot{\leq} c \dot{\leq} c_{i+1} \rightarrow (c_i = c) \vee (c = c_{i+1})] \quad (1)$$

where $\dot{<}$ is the strict order induced by $\dot{\leq}$. The *length* $|p|$ of such a path is m and the set of paths from c_1 to c_2 is denoted by $paths(c_1, c_2)$. One may note that $paths(c_1, c_2) = \emptyset$ each time $c_1 \dot{<} c_2$ is not satisfied. On Fig. 2 the *paths* function always returns a single path since the taxonomy is a tree. Next, the set of all common super-categories of such a couple will be noted as:

$$cap(c_1, c_2) = \{c \in C \mid c_1 \dot{\leq} c \wedge c_2 \dot{\leq} c\} \quad (2)$$

We shall also consider the gap between a couple of categories which satisfy $c_1 \dot{\leq} c_2$ to be the length of the shortest path between them:

$$dist(c_1, c_2) = \min_{p \in paths(c_1, c_2)} |p| \quad (3)$$

For instance, the gap between the singleton category e_1 and the root is 3. The *dist* function is determinant since $dist(c_1, c_2) = 0$ each time $c_1 = c_2$. Finally, the elementary dissimilarity between two categories in the graph is measured as the minimal sum of gaps to a common super-category:

$$\delta(c_1, c_2) = \min_{c \in cap(c_1, c_2)} [dist(c_1, c) + dist(c_2, c)] \tag{4}$$

The exact value of the function $\overline{\delta}(c_1, c_2)$ is obtained through normalisation:

$$\overline{\delta}(c_1, c_2) = \frac{\delta(c_1, c_2)}{\max_{x_1, x_2 \in C \cup C_s} \delta(x_1, x_2)} \tag{5}$$

Here are some examples of what the function computes over the taxonomy of Fig. 2:

$$\begin{aligned} \overline{\delta}(e_1, e_2) &= (3+2)/7 = 5/7, \\ \overline{\delta}(e_1, e_3) &= (3+4)/7 = 1 \ \mathit{and} \\ \overline{\delta}(e_2, e_3) &= (2+4)/7 = 6/7. \end{aligned}$$

It is easy to see that the δ function is a valid dissimilarity measure. Furthermore, when the corresponding taxonomy is a tree (like for TROPES concepts) or a lattice (for ADT), δ is a distance. This also applies to $\overline{\delta}$. They cannot pretend to be tree distances (Barthélémy and Guénoche, 1991) since the ADT structure is not necessarily a tree.

In the following we shall examine the way this function performs in some well known domain structures such as they are generically defined through ADT. Let us first consider a nominal variable. The taxonomy in the classification scheme which models the corresponding domain, say D, is a lattice, isomorphic to the lattice of 2^D for inclusion. Our function follows the complementary of the identity function ($\overline{\delta} = 1 - 1_D$) usually applied in such a case. This fact is easy to detect since, on one hand, the singleton categories correspond to the singleton sets and, on the other hand, the taxonomy contains all the categories corresponding to sets of two individuals. Therefore each $\delta(c_1, c_2)$ such that $c_1, c_2 \in C_s$ is either 0 or 2. Thus, after a normalisation we come to the function indicated previously.
A second case to examine is that of the totally ordered (ordinal) domains. The most frequently used functions over these domains are typically derived from the arithmetic difference. Their common principle may be summarised in the following way: for a couple of values their dissimilarity is proportional to the number of the intermediate values according to the underlying order. Concerning the modelling of such a domain, say D, the classification schemes allow two variants. The first one is identical to what we presented previously so the function $\overline{\delta}$ preserves the same expression. However, this is

rather clumsy. The second one is more clever since it accounts for the order, say $\leq_D$. Its category language L_C is restricted to the set of all intervals over the domain. The resulting taxonomy is still a lattice (a sub-lattice of the one presented above). The singleton categories correspond to the singleton intervals and the supremum of a couple of categories corresponds to the smallest interval to contain both underlying intervals. Thus for a couple of singleton categories c_1 and c_2 whose interpretations are e_1 and e_2 respectively, the supremum $c = c_1 \wedge c_2$ corresponds to the interval $[min(e_1, e_2), max(e_1, e_2)]$. Furthermore, the gaps between both categories and the supremum may be expressed in terms of the corresponding ordinals:

$$dist(c_1, c) = dist(c_2, c) = ord(max(e_1, e_2)) - ord(min(e_1, e_2)) \tag{6}$$

Thus, the elementary dissimilarity between c_1 and c_2 is

$$\delta(c_1, c_2) = 2 \times (ord(max(e_1, e_2)) - ord(min(e_1, e_2))) = 2 \times abs(ord(e_1) - ord(e_2)) \tag{7}$$

and the (simplified) normalised value is

$$\overline{\delta}(c_1, c_2) = \frac{abs(ord(e_1) - ord(e_2))}{ord(max(D)) - ord(min(D))} \tag{8}$$

which is exactly the normalised arithmetic difference over a totally ordered domain.
The third main type of domain structure, the partially ordered one, will not be considered explicitly here. First, this kind of domains does not present any particularity to affect the expression of our function. Thus, the formulae given by the definition are valid and no other (easier) way to compute the function is available. In addition, no function is universally applied over such domains so we have no reference to compare our function with.
This remark about the connection between the topological dissimilarity and the usual measures allows to overload the automatic topological dissimilarity computation in ADT for which it can be computed more quickly. Ae extensive work has also been carried out for considering collection types (like sets and lists) as they are used in TROPES.

If such a measure can be used when a taxonomy is available, it cannot if there is no taxonomy (or, more formally, it is the trivial distance). We define here the topological dissimilarity which measures the resemblance of the individuals only with respect to the taxonomy structures available. For a given product of classification schemes the dissimilarity of a couple of individuals depends only on the elementary differences of the unit projections. Its definition considers two levels in the object structure. The upper level

represents a product of classification schemes (in which, for instance, the taxonomy is under construction) while the lower one is made of a single factor of the product. We consider only direct product factors (no transitivity) and we assume that a taxonomy is available in each of them.
Thus, the dissimilarity of a couple of individuals within the product involves a decomposition of both individuals into their unitary projections. First, let us assume that we have to compute the dissimilarity for K, a product of n classification schemes. Let e and e' be the couple of individuals to compare and e_i and e'_i $(i = 1 \ldots n)$ – the corresponding unitary projections. The dissimilarity between e and e' is given as

$$d_K(e, e') = \sum_{i=1}^{n} \overline{\delta}(e_i, e'_i) \tag{9}$$

The sum operator preserves the properties of non-negativity, symmetry and definitiveness, so d_K is a valid dissimilarity measure too.

5 Classification Algorithm

The topological dissimilarity may be integrated in various taxonomy building algorithms. One of the possible applications is described hereafter.
In TROPES, the set of all concepts and ADT is virtually organised in a dependency graph whose leaves are ADT. Provided the corresponding classification schemes are considered instead, the edges of that graph connect the products to their compounds. The dependency graph is clearly a directed acyclic graph. The algorithm is a composition of several functionality levels:

Build is a depth-first postfix search in the dependency graph which calls **Classify** only when it has been called on the successors (on which the current classification scheme depends) or they already have a taxonomy ;

Classify takes a concept and calls **Partition** on the set of objects and then, on each class obtained, as far as the classes are large enough to be partitionned ;

Partition is a modified version of CLUSTER (Michalski and Stepp, 1983) which does not flatten objects – like CLUSTER/S (Stepp and Michalski, 1986) does – but rather uses the topological dissimilarity on each of the dimensions of the product. The class descriptions are simply constructed through the product operation.

This algorithm deals with compound objects at a reduced computational cost : instead of recursively computing the dissimilarity between each pair

of objects referenced, the dissimilarity depends only on their position in the corresponding taxonomy. It demonstrates the possibility of building a taxonomy on a particular set of objects by using the taxonomy available on the refered objects. One advantage of this approach over the genuine CLUSTER is that the classification process considers the direct fields but does not explore the structure deeper.
Of course, it is impossible to treat in the same way concepts – indirectly – referring to themselves. Some ideas about the way that kind of data can be processed are given in (Bisson, 1995).

6 Conclusion

A topological dissimilarity has been presented which only considers the taxonomy structuring a domain. It allows to compare structured objects regardless of the nature of their components. As a consequence, this measure is able to deal with external data types and structured objects. Such a measure does not ignore the already existing taxonomies. An extension of the CLUSTER/2 algorithm which takes advantage of the dissimilarity has been implemented in the TROPES system. This approach should be extended towards multiple taxonomies (over the same set of objects, see (Euzenat, 1993)) and the consideration of recursive objects.

References

Jean-Pierre Barthélémy and Alain Guénoche (1991): Trees and proximity representations. Wiley, Chichester (GB).

Gilles Bisson (1995) : Why and How to Define a Symilarity Meaure for Object-Based Representation Systems. In: N.J.I. Mars (ed.): Towards Very Large Knowledge Bases. IOS Press, Amsterdam, 236-246.

Cécile Capponi, Jérôme Euzenat, Jérôme Gensel (1995): Objects, types and constraints as classification schemes. Proc. 1st knowledge retrieval, use, storage and efficiency symposium, Santa-Cruz (CA US). to appear.

Edwin Diday (1993): Quelques aspects de l'analyse des données symboliques. Rapport de recherche 1937, INRIA Rocquencourt (FR).

Fioriana Esposito (1994): Conceptual clustering in structured domains: a theory guided approach. In: E. Diday, Y. Lechevallier, M. Schader, P. Bertrand, B. Burtschy (eds.): New Approaches in Classification and Data Analysis. Springer Verlag, Berlin, 395-404.

Jérôme Euzenat (1993): Brief overview of T-TREE: the TROPES Taxonomy building Tool. Proc. 4th ASIS SIG/CR classification research workshop.

Columbus (OH US), 69-87.

Jérôme Euzenat (1995) : Classification schemes: definitions, properties, algorithms. in preparation

K. C. Gowda and Edwin Diday (1994): Symbolic Clustering Algorithms using Similarity and Dissimilarity Measures. In: E. Diday, Y. Lechevallier, M. Schader, P. Bertrand, B. Burtschy (eds.): New Approaches in Classification and Data Analysis. Springer Verlag, Berlin, 414 - 422.

Ryszard Michalski and Robert Stepp (1983): Learning from observation: conceptual clustering. In: Ryszard Michalski, Jaime Carbonell, Tom Mitchell (eds.): Machine learning: an artificial intelligence approach. Tioga publishing company, Palo Alto (CA US), 331-363.

Sherpa project (1995): Tropes 1.0 reference manual. Internal report, INRIA Rhône-Alpes, Grenoble (FR).
`ftp://ftp.imag.fr/SHERPA/rapports/tropes-manual.ps.gz`

Robert Stepp and Ryszard Michalski (1986): Conceptual clustering: inventing goal-oriented classification of structured objects. In: Ryszard Michalski, Jaime Carbonell, Tom Mitchell (eds.): Machine learning II: an artificial intelligence approach. Morgan Kaufmann, Los Altos (CA US), 471-498.