Classification of concepts through products of concepts and abstract data types (abstract)

Petko Valtchev, Jérôme Euzenat

Classification of concepts through products of concepts and abstract data types (abstract)

Petko Valtchev, Jérôme Euzenat

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

Introduction

Object-based systems offer the opportunity of representing and organizing 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, the immediate application of classification algorithms involves a preliminary coding and the reverse decoding stages which may introduce some loss of information [4]. In addition, the data analysis approach, in spite of its undeniable virtues, does not provide a "conceptual" description of the classes [5, 1]. A possible way to avoid these flaws is the use of a conceptual clustering algorithm provided with some further abilities to take into account ordered structures. 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.

We first present the TROPES system and its manner to handle external and primitive types and extend them with a dissimilarity definition. Then, a possible formalization of the taxonomy building process in terms of classification schemes is given. Next, we show the relationship between dissimilarity and class taxonomies. Finally, we give a brief description of a practical classification procedure using concept taxonomies in a bottom-up strategy.

Concepts and ADT in TROPES

TROPES is an object-based knowledge representation model. The individuals are represented as objects. The collection of all objects is 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" where the domain is the concept that object belongs to while the 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 their construction is not accessible to TROPES) are associated with types. Some of them are quite standard (Integers, Booleans...), whereas others are more elaborate (Date, Nucleic acid sequence...). Each of these types is introduced independently from its implementation by an abstract data type (ADT). The ADT for a primitive type T contains an equality predicate between values of this type ($=_T$), a typing predicate ($\in$T) and, sometimes, an order relationship ($\leq_T$).

In order to enable the comparison between values of such types, the ADT are enhanced with a dissimilarity measure. It takes into account the type structure in the ADT, either explicit or unexpressed. For instance, the dissimilarity of two values in a totally ordered discrete type (such as integers) may be defined as the number of intermediate values according to the order ($\leq_T$). A more general dissimilarity ranging over type expression instead of single values may be defined.

The structure of ADT can be mapped into 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 may be enhanced to integrate the dissimilarity. Given a concept K, its extension O_K and a set C_K of classes defined over O_K and organized in a taxonomy (this set is thus structured into a hierarchy $\leq_K$), a dissimilarity measure can be introduced. First, a conventional object dissimilarity lying on the field values may be defined. Second, one may also introduce the class dissimilarity δ_K as an index measured over the hierarchy (δ_K: $C_K \times C_K \rightarrow$„ which is, for instance, some distance inherent to the hierarchy).

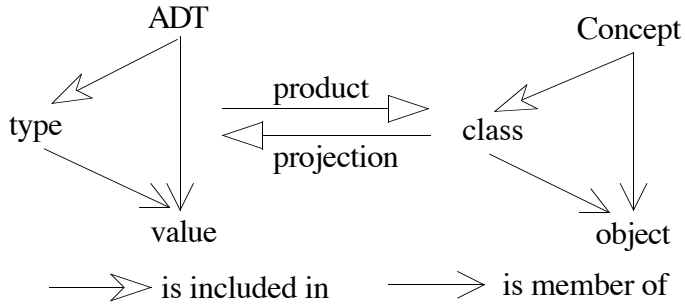

Fig.1. Analogy between concepts and ADT.

How do the above ideas fit in the classification landscape? The compositional vision introduced hereby may be exploited by a classification algorithm. As soon as the sub-structures are provided with their own taxonomies (and the associated dissimilarities) the algorithm should combine them to build a taxonomy in the superstructure. The first attempt to implement similar ideas was an extension of CLUSTER [5] which uses a dissimilarity between two field values, regardless of their nature — ADT values or objects. This way it is possible to process the structure in the initial data with no extra reformulating efforts.

However, the uniform dissimilarity computation is a rather modest gain compared to the tight analogy between concepts and ADT. To take much more advantage of such an analogy, a common framework to clarify the shared structure has to be designed. For that purpose, we use the classification scheme model.

Classification Schemes

A classification scheme is an abstract model of a class taxonomy. 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 the category could 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 category set provided with a partial order. The condition to link the category interpretation to the order relation, called *extension inclusion property* is the following: each time a couple of categories satisfies the partial order predicate a set inclusion between the corresponding interpretations holds. We can now give the definition of a classification scheme which 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 $(C \subseteq L_C)$ and two partial orders: a sub-categorisation relation and a sub-categorisation criterion. 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 and 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 [3].

Classification schemes are constructed from more elementary ones through products. A product is defined over a set of classification schemes in the following 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. We proved [3] the following result: a product of classification schemes is a classification scheme. The reverse operation, called projection, is defined too. It 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. This way, ADT in TROPES are elementary classification schemes where both orders coincide. Concepts are, in turn, products of classification schemes representing ADT or other concepts.

The main advantage the classification scheme model provides is the homogeneous manner the different sub-structures of a concept are accounted for. 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 compounds are identified under the corresponding more elementary 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. Our idea was to apply the same principles of overall homogeneity to the automatic classification.

Dissimilarity

The complete integration of automatic classification algorithms into the above model requires that model to be extended. We need it, at first, to integrate such a fundamental notion as the dissimilarity between individuals.

In other words we should find a generic dissimilarity measure to hold within a classification scheme. It means that the new measure should be based exclusively upon the taxonomy structure. In addition, we would like it to cover some well-known measures used in particular situations. Those requirements taken into account, we designed what we called the "topological distance". The basic idea is to explore, for a product, the taxonomy structures within the classification schemes-compounds. The dissimilarity between a couple of individuals within the product is the total of their unitary projection dissimilarities. The point is that those projection dissimilarities are computed over the corresponding taxonomy. An exact description of the way the new dissimilarity is computed follows.

First, let us assume that we have to compute the dissimilarity for K, a product of n classification schemes all of them provided with a taxonomy. Let now compare two individuals, say e and e'. Let the corresponding unitary projections be e_i and e_i' (i = 1..n) respectively. The dissimilarity between e and e' is given as

$$d_K(e, e') = \Sigma_{i=1..n}\, \delta(e_i, e_i').$$

The substantial role in the above formula is played by the δ function, computed in the same way over each projection. We set it to measure the gap between its parameters over the graph generated by the taxonomy[1]. The value of δ is proportional to the shortest paths from both parameter nodes to their most specific common category. Actually the mean of those path lengths is taken. The final value is obtained after a normalisation using the greatest chain length within the graph.

Our definition is valid each time a unique most specific common super-category for arbitrary couple of categories in the taxonomy exists. In other words, we need at least an upper semi-lattice structure to apply the above measure. Otherwise an extension of the latter to compute the least paths length sum among all possible has to be applied.

Another question one may rise concerns the features of the defined dissimilarity. It is easy to see that the δ function is a valid dissimilarity measure. In fact, it is positive, symmetric and it equals 0 each time both parameters are identical. The sum operator preserves those properties, so d_K is a valid measure too. Furthermore, when the corresponding taxonomy is a tree, δ is a distance function since it respects the triangle inequality.

Classification Algorithm

The previously presented 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 (concept) dependency graph. Provided the corresponding classification schemes are considered instead, the vertices of that graph connect the products to their compounds. The dependency graph is clearly an oriented and circuit-free one.

The algorithm is a composition of several functionality levels. The top level explores the dependency graph from a chosen classification scheme downwards. This is made in a depth-first post-fix manner, testing at each node the availability of a taxonomy within the corresponding classification scheme. The nodes that score positively for that test (these are

---

[1] To construct that graph we extend the taxonomy by connecting the individuals to their most specific categories. This way we obtain an oriented circuit-free graph.

typically ADT and some elementary concepts) receive no further treatment. They draw the limits of the exploration, providing in the same time the basis for the automatic classification in higher nodes. At each of the remaining nodes reached by the algorithm the exploration carries on with the successors in the graph. Once all of those successors has been thoroughly processed, that is a taxonomy is available (either already existing or built within a previous step) the automatic classification of the current node may start. It constitutes a second level of our algorithm. The underlying procedure combines the taxonomies over the substructures of the concept by the product operation to obtain a prototype of the future taxonomy to validate. The dissimilarity measure between the objects and the proto-categories is used to obtain a simpler and more "plausible" taxonomy. It allows only the high-scored proto-categories to be retained. The resulting taxonomy, in turn, induces a new dissimilarity measure on the set of the concept objects.

As we noted above, we have been working on CLUSTER to make that algorithm stick with an objet-based model like TROPES. Later on, the underlying star algorithm showed to be fairly well adapted to the classification scheme formalism and the product/projection duality. So we came to a extension of CLUSTER based upon the new metrics and the notion of classification scheme product. That modification is what we currently use as a taxonomy building routine in the above algorithm.

What are the advantages of our algorithm from the (conceptual) clustering viewpoint? First, our algorithm deals with compound objects. That kind of treatment is offered 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. Another advantage of this approach is the availability of immediate category descriptions through the classification scheme product operation. We thus combine a "conceptual" clustering with an extendible type system. Next, objects are treated with their original structure. Compared with CLUSTER/S, a typical algorithm exploiting the structure in the data, we use no flattering stage.

At that moment we can draw a list of possible extensions our approach would support. One of them concerns the classification with regard to several taxonomies over the same concept. Each taxonomy in this case constitutes what we call a viewpoint. The viewpoints are assigned a classification scheme each and it is often profitable to treat those classification schemes in a single global one. Another extension we think about concerns the type constructors. So far we have considered only single-valued fields. In TROPES, however, the type system supports multi-valued attributes. The latter are realised through constructors like lists and sets. Processing correctly lists and sets attributes is mandatory for any classification algorithm to fully integrate the type system in TROPES.

There are some limitations to our approach, and namely, the possibility of having concepts — indirectly — referring to themselves is not dealt with. Some ideas about the way that kind of data can be processed are given in [7].

References

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

[2] K. C. Gowda, Edwin Diday, 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, pp414 - 422. Springer Verlag, Berlin, 1994

[3] Jérôme Euzenat, Une définition abstraite de la classification et son application aux taxonomies d'objets, proc. 2ndes journées représentation par objets (RPO), La Grande Motte (FR), pp235-246, 1993

[4] Jérôme Euzenat, Brief overview of T-TREE: the TROPES Taxonomy building Tool, proc. 4th ASIS SIG/CR classification research workshop, pp69-87, Columbus (OH US), 1993

[5] Ryszard Michalski, Robert Stepp, Learning from observation: conceptual clustering, in Ryszard Michalski, Jaime Carbonell, Tom Mitchell (eds.), Machine learning: an artificial intelligence approach, pp331-363, Tioga publishing company, Palo Alto (CA US), 1983

[6] Gilles Saporta, Probabilités, analyse de données et statistique, Editions Technip, 1990

[7] Gilles Bisson, Why and How to Define a Symilarity Meaure for Object-Based Representation Systems, in N.J.I.Mars (ed.) Towards Very Large Knowledge Bases, pp236-246, IOS Press, Amsterdam,1995