Applying DLs for Retrieval in Case-Based Reasoning *

Pedro A. González-Calero, Belén Díaz-Agudo and Mercedes Gómez-Albarrán
Dep. Sistemas Informáticos y Programación
Universidad Complutense de Madrid, 28040 Madrid. Spain
email:{pedro, belend, mercedes} @sip.ucm.es

1 Introduction

Case-Based Reasoning (CBR) is a problem-solving paradigm that uses knowledge of relevant past experiences (cases) to interpret or solve a new problem case [7]. During the last few years, Description Logics (DLs) have become systems of great interest for the Case-Based Reasoning community. There has been a body of work [3, 5, 6, 8, 11, 13] on taking advantage of the DL reasoning mechanisms for the processes involved in the CBR cycle.

Although a number of different approaches are considered, they all focus on the intuition that the formal semantics and the capability of DLs to maintain a terminological taxonomy are interesting properties to manage a case base. Moreover, DL declarative semantics helps in the domain comprehension; their ability to automatically classify concepts and recognise instances is a useful property for the case base management; the incoherence detection mechanism helps in the domain model development and the consistency checking; and their ability to build structured case descriptions provide a flexible and expressive way to represent the cases.

The aim of this position paper is the description of a general and domain-independent approach taking advantage of the DL capabilities for the case similarity assessment and retrieval in a CBR context.

2 Using DLs for retrieval in CBR

The main knowledge source in a case-based reasoner are the specific pieces of experience which have been represented and stored to be reused in the future. We propose the use of DLs to represent all the knowledge needed in a case-based reasoner. The DL knowledge base ($\mathcal{KB}$) contains the specific knowledge represented by the cases and the general domain knowledge that is included to support the CBR processes.

As in [6, 8], we represent a case as an individual. Each case is linked by DL relations to two individuals that represent the *description* of the problem and the *solution* that was carried out to solve it.

In the line of [5], we distinguish different approaches to implement retrieval using the reasoning mechanisms available in DLs:

- The *deductive approach* bases the case retrieval on *concept classification* [6, 8]. A concept description is built using the restrictions specified in the query. This concept is then classified, and finally all its instances are retrieved.
- The *inductive approach* relies mainly on *instance recognition* [11, 13]. An individual is built and a number of assertions are made about it based on the features specified in the query. Instance recognition is applied to retrieve the most specific concepts of which this individual is an instance, and then all the instances of these concepts are retrieved.

There are two main differences between them:

- The type of restrictions that can be included, since concept description language and assertion language are different. Concept description language is richer because restrictions about role type and cardinality can be included.
- Instance completion. There are a number of inferences that are only applied to individuals. DL systems do not enrich concept descriptions with inferred constraints, but just take the concept definition as it is, and classify it accordingly. On the other side, when an individual is recognized as instance of a given concept, based on the sufficient conditions for belonging to that concept, then necessary conditions on the concept definition are automatically asserted on the individual.

So, both approaches have its pros and cons, namely expressiveness vs. completion. We have decided to implement retrieval as an instance recognition process mainly for one reason: instance completion accomplishes a kind of query completion where additional constraints can be

*This work is supported by the Spanish Committee of Science & Technology (CICYT TIC98-0733)

automatically inferred from those explicitly asserted by the user. In this way, we have a straightforward method to let the domain knowledge assist in the crucial process of query formulation.

Using the domain terminology of the $\mathcal{KB}$, the user describes the query. From this information the system builds an instance and recognises it. Once the system has retrieved the individuals representing the case descriptions that are instances of the most specific concepts the query is instance of, they are ranked by its similarity with respect to the individual representing the query. For this purpose we apply a domain-independent similarity function along with a number of domain-specific heuristics.

3 Similarity measure

We are interested in computing the similarity between a given pair of individuals: one representing the query, and another representing the candidate case description. An individual is defined in terms of the concepts of which is an instance and the roles explicitly asserted for it or entailed by the contents of the $\mathcal{KB}$. In DLs, the roles of an individual are represented as relations connecting the individual to other individuals or primitive values (*fillers*).

A key decision in our approach to similarity measure stems from the fact that, in general, the information about an individual is not completely defined by the concepts of which that individual is an instance, since: additional roles may appear in the individual; concept definition may constraint role cardinality to be within a given range, but in the individual a fixed number of fillers is given; concept definition may constraint the type of role fillers, but in the individual actual fillers appear which may further specialize the type restriction. These differences recursively extend since role fillers may in their turn be individuals, which carry more information than the concept used to constraint the type of fillers of a given role, if any. Therefore, our similarity measure separately computes the contribution of the concepts and the contribution of the actual roles appearing in the individual. This decision distinguishes our work from previous approaches.

It may be argued that the separate computation of concept and role based similarity is redundant, since the superconcepts of an individual are influenced by the role assertions of the ABox. Redundancy could be avoided if role-based similarity would be restricted to those additional roles which are not constrained in concept definitions. This decision, however, would result in a lost of information, since actual fillers may be more informative than role type/cardinality restrictions. Yet another solution could be to consider only primitive concepts when computing the concept-based similarity. The reason for not adopting this solution is that every defined concept explicitly included in the $\mathcal{KB}$ represents a meaningful abstraction, and, therefore, the very existence of that concept should affect the similarity function.

The second key feature of our approach is that we intend to minimize the human effort required in building the similarity framework. By taking advantage of the structure of the $\mathcal{KB}$, we try to reduce the number of explicitly hand-coded similarities.

Concept-based similarity

The concept-based similarity between two individuals will be given by the similarity between the two sets of concepts of which those individuals are instances. We can simplify the problem by selecting only the most specific concepts from both sets, since a concept carries all the information contained in its superconcepts. A further simplification is to consider, as the most on point concept for describing an individual, the one obtained as the conjunction of the most specific concepts of which that individual is an instance, which may or may not exist in the TBox. This way we reduce the problem of comparing two sets of concepts to that of comparing two single concepts.

Evaluating relatedness using network representations is a problem with a long history in Artificial Intelligence. A natural way to evaluate semantic similarity in a taxonomy is to evaluate the distance between the nodes corresponding to the items being compared—the shorter the path from one node to another is, the more similar they are. A problem with this—intensional— approach, however, is that it relies on the notion that links in the taxonomy represent uniform distances.

An alternative—extensional—approach can be applied when nodes in the taxonomy represent classes of individuals, which are also available in the system, as is the case in DLs with concepts and its instances. In this case, the similarity between two concepts can be obtained, basically, by the number of its common instances. This is the approach described in [10], extended with ideas from information theory. Probabilities are associated with concepts in the taxonomy, such that given a concept c, $p(c)$ is the probability of encountering an instance of concept c, computed as the relative frequency of the instances of c in the $\mathcal{KB}$. The similarity between concepts is measured as the information they share, given by the maximum of information content of the most specific concepts that subsume them in the taxonomy. Finally, the information content of a concept is quantified as $-log\ p(c)$.

The problem with the *extensional approach* is that it relies on a basic assumption, which may or may not apply for a given $\mathcal{KB}$, namely, that concepts are evenly populated with instances. A non-homogeneous distribution of individuals among concepts will bias the similarity measure, since the number of instances determines the information content.

Notice that both—intensional and extensional—approaches present a nonmonotonic behaviour; different results will be obtained for the similarity between two given concepts as concepts and/or individuals are added or removed from the $\mathcal{KB}$. This situation may be considered as a feature or a problem, depending on the particular application. Anyway, this is more relevant to the *extensional approach* since, once delivered, the TBox tends to be much more stable than the ABox.

For concept-based similarity we adopt an intensional approach where only the information available in the TBox is considered. We have applied the *vector space model* used in Information Retrieval [12]. In this model, every indexable item is represented by an attribute vector, and the similarity between two items is obtained by some kind of algebraic manipulation of the vectors associated with them.

We consider as attributes the concepts defined in the $\mathcal{KB}$. Formally, given the set $C = \{c_1, \ldots, c_N\}$ of concepts defined in a $\mathcal{KB}$, for each concept $c_i \in C$ we define a vector v_i such that its component k equals 1 if c_k subsumes c_i, and 0 otherwise.

Once the vectors have been built, the *conceptual similarity* between two concepts is computed as the cosine of the angle formed by the vectors representing them, a similarity function usually applied in the vector space model:

$$\mathrm{sim}(c_i, c_j) = \frac{v_i \cdot v_j}{\|v_i\| \cdot \|v_j\|}$$

In [4] it is proved that this expression is equivalent to the following one, the one that we really use to compute the similarity among concepts:

$$\mathrm{sim}(c_i, c_j) = \frac{|\bigcap \mathrm{super}(c_i, C)\mathrm{super}(c_j, C)|}{\sqrt{|\mathrm{super}(c_i, C)|} \cdot \sqrt{|\mathrm{super}(c_j, C)|}}$$

where $\mathrm{super}(c, C)$ computes the set of concepts in C which are superconcepts of c. So, the similarity between two concepts is given by the number of their common superconcepts and their total number of superconcepts, which can be dynamically computed, making unnecessary the explicit representation of attribute vectors.

Role-based similarity

The basic idea of role-based similarity is that two individuals are more similar when the graph induced on them by role assertions is more similar. In general, role assertions may have been explicitly asserted or may be entailed by the TBox and ABox. We consider comparable only those roles with the same relation, and compute the similarity between roles as the similarity between its fillers. Since relations are in general multiple valued, role comparison turns out to be a comparison between sets of individuals.

When comparing sets of individuals we recursively apply the function of similarity between individuals, accumulating for every individual of one set the maximum of the results obtained by comparing that individual with every individual of the other set. A more pessimistic approach could be taken by computing for every individual of one set the average similarity between that individual and every individual of the other set, or even the minimum of those values. This recursion ends when comparing individuals without slots, whose similarity is given by the concept-based similarity term. Formally, given the set O of individuals defined in a $\mathcal{KB}$, the similarity between O_1 and O_2, $O_1, O_2 \subseteq O$, is given by the expression shown in Figure 1.(a), where the min of a couple of sets is the set with the minimum cardinality, and the max, the set with the maximum cardinality.

Putting the pieces together

Finally, the similarity between two individuals is computed as the sum of the similarity between the—conjunction of—most specific concepts of which those individuals are instances and the similarity among their roles, as is formally defined in Figure 1.(b), where the set R denotes the set of relations defined in the $\mathcal{KB}$ and the functions $\mathrm{t}(o)$ and $\mathrm{v}(o, r)$ obtain the most specific concept of which the individual o is an instance, and the set of individuals connected with the individual o through the relation r, respectively.

As the alerted reader may have noticed, problems may arise with the role-based similarity function when cycles appear in the ABox. Two solutions can be considered in that case: to collect the visited individuals as the algorithm proceeds, so that none is visited twice; or, the mechanism we use, to identify the relations which may produce circularity—such as those defined as the inverse of another relation—so that those roles are never compared. Potentially circular relations may be easily identified by making them specializations of the *circular-relation* defined for that purpose.

The similarity measure described so far is domain-independent and can accurately take into account the structure of the $\mathcal{KB}$, without further human intervention. Nevertheless, in order to make it more applicable, our framework allows for the integration of domain-specific similarity knowledge in one of the following ways:

Concept-specific similarity knowledge This kind of restriction let specify that instances of a given concept should be compared by a subset of their roles. This way, descriptive roles can be distinguished from those others that provide additional

$$\mathrm{sim}(O_1,O_2) = \begin{cases} 0 & \text{if } O_1 = \emptyset \vee O_2 = \emptyset \\ \dfrac{\displaystyle\sum_{o_i \in \min(O_1,O_2)} \max_{o_j \in \max(O_1,O_2)} (\mathrm{sim}(o_i,o_j))}{|\min(O_1,O_2)|} & \text{otherwise} \end{cases}$$

(a) Similarity function between two sets of individuals

$$\mathrm{sim}(o_1,o_2) = \begin{cases} \mathrm{sim}(\mathrm{t}(o_1),\mathrm{t}(o_2)) & \text{if } \forall r \in R : \mathrm{v}(o_1,r) = \emptyset \wedge \mathrm{v}(o_2,r) = \emptyset \\ \frac{1}{2}\left(\mathrm{sim}(\mathrm{t}(o_1),\mathrm{t}(o_2)) + \frac{\sum_{r_i \in R} \mathrm{sim}(\mathrm{v}(o_1,r_i),\mathrm{v}(o_2,r_i))}{|\{r \in R | \mathrm{v}(o_1,r) \neq \emptyset \vee \mathrm{v}(o_2,r) \neq \emptyset\}|}\right) & \text{otherwise} \end{cases}$$

(b) Similarity function between two individuals

Figure 1: Similarity functions

information, but should not be considered when determining the similarity between two individuals.

Relation-specific similarity knowledge This kind of restriction let specify that an alternative function should be applied when computing the similarity between the fillers of a given role (i.e., the values of a given attribute). For example, this mechanism could be used to specify the similarity between values of primitive types by specifying the function to be applied on those relations that are to be filled by primitive values. This situation is specially relevant when comparing slotless individuals, since if they are instances of the same concepts, the domain-independent measure will always compute a maximum similarity of 1 for them.

A high level mechanism combines the different similarity measures, trying first to obtain domain-specific measures and if none applies, computing the domain-independent function.

4 Related work on similarity measures

We have identified two approaches to measure the similarity between the cases. The first one computes a *numerical value* either using the number of shared properties [6, 13] or the hierarchical distances between concepts [1, 2]. In [1] the *intersect* function—similar to the Least Common Subsumer (LCS)—is used to build a concept whose description is the abstraction of the properties shared by two cases. Case selection computes a numerical value according to the place in the hierarchy of this concept.

The second—declarative—approach uses *symbolic criteria* to represent the similarity between cases. In [9, 11] an approach similar to [1] is presented but, instead of computing a numerical value, the subsumption relationship between the concepts representing the similarity—and dissimilarity in [11]—is used to order, compare and select the cases that are instances of the most specific similarity concepts. Additionally, the explicit similarity representation, makes easier to convince the users of the plausibility of the system's conclusions, because the system can justify, using the domain terminology, in which sense the precedent matches the query.

In [13] a semantic-based measure for similarity that reasons and analyses the meaning of concepts is defined. The weakness is that, although the differences between concepts are based on the DL concept forming expressions —what means that uses more of the DL paradigm—only value restrictions and number restrictions are considered. Moreover the only studied case is when a unique difference exists between the concepts being compared. Besides, the algorithm to calculate the similarity is not completely detailed.

As ours does, the work presented in [2] also takes advantage of the knowledge about the similarity contained on the class hierarchy of an object oriented case representation.

The similarity is divided into the intra-class similarity SIM_{intra} and the inter-class similarity SIM_{inter}, that are multiplied to compute the final similarity between the query object and the case object. SIM_{intra} is based on the common attributes, i.e. on the attributes of the least common class of the two objects being compared, and SIM_{inter} represents the similarity of two objects depending only on the positions of their classes in the hierarchy. That way, the class hierarchy encodes some knowledge about inter-class similarity between the real-world objects: the deeper the more specific common class between the objects being compared is, the greater their inter-class similarity is. To give a numerical value to this property they propose to add additional knowledge to the hierarchy by manually annotating every inner node with a similarity value that is greater for deeper nodes. The main advantage of this approach is an inexpensive run-time retrieval computing at the expense of previously annotating the nodes. Moreover, our proposal is automatic and can be computed in a dynamical way taking into account the structure of the $\mathcal{KB}$ avoiding the human intervention.

5 Conclusions and future work

In a CBR system, the case representation and organization should provide the means to index, retrieve and compare cases in an efficient and reliable way.

In this position paper we have highlighted the utility of DLs as a suitable formalism to represent and organize the knowledge used during the case retrieval.

We have defined a retrieval process and a domain independent numerical similarity measure, where case structure and $\mathcal{KB}$ organization is accurately taken into account. We have also shown how the domain-independent similarity measure can be integrated with domain-specific similarity knowledge.

A requirement is that the concept hierarchy should be balanced in depth. The general purpose similarity function takes into account not only shared features but also the total number of features an individual has. So, those individuals which are deep in the hierarchy—i.e., have more features—will never be chosen if individuals, with a similar—or even smaller—number of shared features with the query, exist higher in the hierarchy. This situation should be avoided when it is the result of an unbalanced knowledge representation, where a portion of the domain has been described in more detail—with more levels of abstraction—than others, and individuals from different parts of the $\mathcal{KB}$ are to be compared.

As a future work, a mixed intensional and extensional approach could be used by using the information content measure to assign weights to the concepts in the vector space model.

Nowadays, we focus on the case adaptation process. We are developing a substitution based adaptation mechanism guided by the explicit representation of dependency relations in case solutions.

Although we have not accomplished a formal and detailed evaluation of the similarity function, we have shown the applicability of the proposed model by developing OoFRA, an effective CBR assistant for object-oriented framework reuse [3]. Nevertheless a more exhaustive evaluation should be made. A possible strategy for this evaluation [10] could be to make human experts in a given domain rank the similarity among a number on individuals and then apply the similarity function and do some kind of regression test.

Acknowledgements

We would like to thank the reviewers for their insightful comments, which have enhanced the clarity of this position paper.

References

[1] Beck, H.W., 1991: "A conceptual clustering algorithm for semantic data model", *International Workshop on Terminological logics.*

[2] Bergmann, R., Stahl, S., 1998: "Similarity Measures for Object-Oriented Case Representations", *Procs. EWCBR'98.*

[3] Gómez-Albarrán, M., González-Calero, P., Díaz-Agudo, B. & Fernández-Conde, C., 1999: "Modelling the CBR Life Cycle Using Description Logics", *Procs. ICCBR'99.*

[4] González-Calero, P.A., 1997: "Aplicación de técnicas basadas en conocimiento como soporte a la reutilización en bibliotecas orientadas a objetos", PhD Dissertation, Departamento de Informática y Automática, Universidad Complutense de Madrid.

[5] Kamp G., 1996. "Using Description Logics for Knowledge Intensive Case-Based Reasoning", *Procs. EWCBR'96.*

[6] Koehler J., 1994: "An Application of Terminological Logics to Case-based Reasoning", *Procs. KR'94.*

[7] Kolodner, J., 1993: *Case-Based Reasoning*, Morgan Kaufmann.

[8] Napoli A., Lieber J. & Simon A., 1997: "A Classification-Based Approach to Case-Based Reasoning", *Procs. DL'97.*

[9] Plaza, E., 1995: "Cases as Terms: A feature term approach to the structured representation of cases", *Procs. ICCBR'95.*

[10] Resnik, P., 1995: "Using Information content to Evaluate Semantic Similarity in a Taxonomy", *Procs. IJCAI'95.*

[11] Salotti S. & Ventos V., 1998: "Study and Formalization of a Case-Based Reasoning System using a Description Logic", *Procs. EWCBR'98.*

[12] Salton, G. & McGill, M. J., 1983: *Introduction to Modern Information Retrieval*, McGraw-Hill.

[13] Yen J., Teh H., & Liu X., 1994: "Using Description Logics for Software Reuse and Case-Based Reasoning", *Procs. DL'94.*