Cases as terms: A feature term approach to the structured representation of cases

Enric Plaza
Institut d'Investigació en Intel·ligència Artificial, CSIC
Campus de la Universitat Autònoma de Barcelona
08193 Bellaterra, Catalunya, Spain
Email: enric@iiia.csic.es
URL: http://www.iiia.csic.es

1 Motivation

In our research work, we have come to represent cases as complex, structured data structures that we will formally describe as feature terms (see [Arcos] for a description of the NOOS language for CBR and multistrategy learning). The advantage of using structured representations are twofold. Firstly, it offers a natural way to describe composite objects that if described by attribute-value representations cause some problems like dealing with irrelevant attributes, not-applicable values, etc. (and this leads to problems when comparing similitude among cases in this descriptions).

Secondly, structured-representation cases offer the capability of treating subparts of cases also as full-fledged cases: they can be stored, and retrieved and used to solve (sub)problems of new cases; also, a case may be solved using (subparts of) multiple cases retrieved from the system's memory.

We will present here a formalization of structured (sometimes called object-centered) representations as feature terms and we will present how can we assess similarity between feature terms (cases) and determine the preferred (most similar) case from a set of cases.

1.1 Background

This approach is different from some usual notions of case-based reasoning (CBR). Usually, cases are (represented as) tuples of attribute value pairs for a known, finite vocabulary of attributes and values (except for numeric values that need not be finite). Then, similarity is estimated using some metrics (that may involve differential weighting of attributes, dynamic adjustment of the weights, etc.).

The usual thing is to have some attribute-wise distance measure, i.e. if attributes are a_i and we use the notation $t.a$ to denote the value of attribute a in case t, then attribute-wise distance is some function δ that may be defined according to the a_i attribute's type. Then some aggregation function D is defined as the weighted distance of two cases t_1, t_2. There are some problems that have to be solved, like missing values, irrelevant values, etc. Next, some similarity function is defined as S where N is a normalization function.

Other CBR approaches do not use similarity but symbolic indices (sometimes learned by explanation-based generalization). However, this approach deals only with what we call [Armengol] retrieval task, and not with the selection task of the retrieved cases [Plaza]; usually these CBR systems use some similarity-based or domain-specific strategy for ranking the retrieved cases.

We will see how our approach deals with both retrieval and selection of structured cases. Indeed, the structured representation of cases we propose has the advantage of a wider expressive

power than attribute-value representations. However, the classical approach of distance among tuples is not directly applicable. One option is that of extending distance metrics to structures representations as trees (see [Bunke]).

1.2 Towards similitude descriptions

Our approach is slightly different. Let's ask ourselves the reason of using distances: it is a way of estimating similarity between cases, and we want this because later the selection process in the CBR system will pick up as most relevant the most similar case.

What we need then is a method for assessing similarity between cases (represented as structured terms) and obtaining a preference ranking among the cases based on that similarity. We do not need to use a distance (although it provides a complete ordering) if we have a way to obtain a preference ordering (albeit a partial ordering) based on a similarity estimate.

In the following we will formalize structured representation of cases as feature terms and we will show how this formalization allows us to define a natural characterization of similarity and how it can be used in the retrieval and selection of cases.

This formalization is general and hopefully useful for other CBR systems, but we will explain specifically how NOOS, the language for CBR and multistrategy learning we have developed, is able to represent and implement all the concepts and methods we introduce. Feature terms can thus be considered as the foundation of CBR representation in NOOS-based applications [Arcos].

The structure of the paper is as follows: first feature terms are introduced and formalized, next the notions of subsumption and antiunification of feature terms are defined. Antiunification provides a description of the similitude among terms (cases). These notions are then used to realize a strategy for selection of retrieved cases using on a preference ordering based on a ranking of similitude descriptions.

2 Feature terms

Before defining formally feature terms, we will introduce intuitively some basic notions. A structured representation of cases is one where the value of an attribute of a tuple (case) is another tuple. This fact allows to treat subparts of cases as full-fledged cases.

The intuition behind feature terms is that the basic power of this representation is that of path equality. A path is a concatenation of attributes. Path equality is a restriction upon the values the term's attributes appearing in two paths may take. For instance a path equality constrains the term's attributes to have the same value. That is to say, they may have any value as far as among them the following relationship holds: following those paths they lead to the same term.

In fact, feature terms are just a generalization of first order terms that are useful to have a declarative representation of record-like structures without the loss of the convenient instantiation ordering unification operation [Aït-Kaci].

Using first order terms we can represent a person in this way: `person(John, Smith, 34, Jack-Smith, NYCity)` where we have to know the meaning of the arguments by its position. Using feature terms each argument is denoted by a symbol identifier instead of position. This change allows to easily express incomplete information. We could not do that using the above first order term for person.

For instance in a form of feature terms called unification grammars, the syntax for a person whose father is unknown can be represented. The same term in NOOS syntax (and giving it the name John-Smith) is expressed:

```
(define (person John-Smith)
    (name (define (name)
```

```
        (first John)
        (last Smith)))
(lives-at (define (address)
            (city NYCity)))
(father (define (person)
        (name (define (name)
                (last (>> name last)))))))
```

where path equality is expressed in position `John-Smith.father.name.last` by `(» name last)` that determines a path starting at the root (i.e. John-Smith) and followed by name and last features.

2.1 Feature terms formalization

Regardless the concrete syntax used in feature terms, the abstract syntax of feature terms can be defined as follows (we will use [Carpenter] formalization, an alternative formalization may be found in [Aït-Kaci]). Let us have a finite set $Feat$ of features (attributes) and a type hierarchy $\leq$ that is a finite bounded complete partial order over the type symbols of the set $Type$.

Definition 2.1 (Feature term)
A feature term $F = \langle Q, q_r, T, D \rangle$ over $Type$ and $Feat$ is a tuple where:

- Q : a finite set of nodes rooted at q_r,
- $q_r \in Q$: the root node
- $T : Q \rightarrow Type$: a total node typing function
- $D : Feat \times Q \rightarrow Q$: a partial feature value function

Let $\mathcal{F}$ denote the collection of feature terms. The intuitive way to picture a feature term is as directed labelled graphs such that Q is the set of nodes, T determines the labels of the nodes, and arcs are determined as follows: there is an arc from q to q' labelled by f, written $q \xrightarrow{f} q'$, if $D(f, q) = q'$.

Definition 2.2 (Paths)
A path π is a sequence of features. Let $Path = Feat^*$ and ϵ the empty path, we can now extend the feature value function D to paths as follows:

- $D(\epsilon, q) = q$
- $D(f\pi, q) = D(\pi, D(f, q))$

The root of a feature term is the node q_r that satisfies $Q = \{D(\pi, q_r) \mid \pi \in Path\}$, i.e. the node from which all nodes in Q are reachable. It is then apparent that defining a feature term is equivalent to assigning feature terms as the values of paths.

Definition 2.3 (Path Value)
If $F = \langle Q, q_r, T, D \rangle$ and $D(\pi, q_r)$ is defined, then $F \setminus \pi = \langle Q', q'_r, T', D' \rangle$ where:

- $q'_r = D(\pi, q_r)$
- $Q' = \{D(\pi', q'_r) \mid \pi' \in Path\}$
- $D'(f, q) = D(f, q)$ if $q \in Q'$
- $T'(q) = T(q)$ if $q \in Q'$

There are two interesting properties of feature terms. The first one is that all subterms of a feature term (that are also feature terms) are univocally determined by the path leading to it from the root. A second property of feature terms is that two terms may be equivalent up to the renaming of their nodes if they represent the same information. This will be useful for defining the subsumption of feature terms.

The intuitive idea is considering feature terms as descriptions of situations, then subsumption is the usual one: a description F subsumes another description F' if F is more general than F'. Regarding feature terms, the essential idea is that a feature term F subsumes F', written $F \sqsubseteq F'$, if and only if (1) the type of the root of F' is a subtype in $\leq$ of the root of F, (2) all features defined in F are also defined in features of F' (and the feature values of F subsuming the corresponding feature values of F'), and (3) all path equality constraints in F also hold in F'.

Notice that we treat subsumption ordering $\sqsubseteq$ as an informational ordering, i.e. $a \sqsubseteq b$ means that a has less or equal information content than b (a is more general and b more specific). In this approach the result of a join operation ($\sqcup$) of two descriptions is a description that has the union of the information, while the result of a meet operation ($\sqcap$) of two descriptions is a description that has the information in common to both descriptions.

Definition 2.4 (Subsumption)
$F = \langle Q, q_r, T, D\rangle$ subsumes $F' = \langle Q', q'_r, T', D'\rangle$, $F \sqsubseteq F'$, if and only if there is a morphism $h : Q \to Q'$ such that:

- $h(q_r) = q'_r$
- $T(q) \sqsubseteq T'(h(q))$ for every $q \in Q$
- $h(D(f, q)) = D'(f, h(q))$ for every $q \in Q$ and feature f such that $D(f, q)$ is defined.

Lemma 2.1 (Feature term pre-ordering)
The subsumption relation $\sqsubseteq$ is a pre-ordering of the collection $\mathcal{F}$ of feature terms. When two feature terms subsume each other ($F \sqsubseteq F'$ and $F' \sqsubseteq F$) they are identical up to a renaming of nodes ($F \equiv F'$).

2.2 Antiunification

The notion we will use for assessing similarity of structured case representations, formalized as feature terms, is that of antiunification. Intuitively, the antiunification of two feature terms gives that which is common to both (yielding a generalization) and all that is common to both (the most specific generalization). In formal terms, the antiunification of two feature terms, $F \sqcap F'$, yields the greatest lower bound with respect to subsumption ordering.

Procedure AU(F, F')

- Let R and R' be the types of the root nodes of F and F'
- The root of the new term is of type $meet(R, R')$
- For all features f defined in both F and F'
  - Let V and V' be the feature values of F and F' for feature f
  - If V and V' are involved in a path equality constraints in both F and F' and they have already been antiunified in a feature f'

    Then the value of feature f in the new term is the value of f'

    Otherwise Let $W = AU(V, V')$
  - If $W = \bot$ then f is not a feature of the new term
  - Otherwise f is a feature of the new term with value W

Definition 2.5 (Antiunification)
Let $F, F' \in \mathcal{F}$ be feature terms such that $F \equiv \langle Q, q_r, T, D \rangle$ and $F' \equiv \langle Q', q'_r, T', D' \rangle$ are such that $Q \cap Q' = \emptyset$. We define an equivalence relation R on $Q \times Q'$ as a binary relation such that:

- $R \subseteq Q \times Q'$
- $R(q_r, q'_r)$
- $R(D(f, q), D'(f, q'))$ if and only if both are defined and $R(q, q')$.

The antiunification of F and F' is then defined to be $F \sqcap F'$.

3 Retrieval and selection of cases

We will show in this section how these notions can be used for reasoning about structured cases.

3.1 Similitude terms

With antiunification we can build descriptions of the structural similitude between two feature terms: $AU(t_1, t_2) = t_1 \sqcap t_2$, and let us call it a similitude term of t_1 and t_2.

What a CBR system needs now is a way to assess which case of a case base CB is more similar to a given case. Since we represent all cases by feature terms, we can build the similitude term set S_{t_0} of a case t_0 in a case-base CB as follows:

$$S_{t_0} = \{AU(t_0, t_i) \mid t_i \in CB\}$$

Since similitude terms are also feature terms, they are in $\mathcal{F}$ and we can use the pre-ordering to reason about them. We can map the $\sqsubseteq$-ordering over the similitude term set S_{t_0} into a preference ordering upon the CB. The most preferred cases in CB are the maximals with respect to the similitude term set.

3.2 Using feature terms in NOOS case-based applications

In this section we will show some examples of using domain knowledge for retrieval and selection of cases in developing a specific CBR application. For illustration, we will use CHROMA, a NOOS-based system.

In NOOS we use feature terms (graphs) as queries for retrieval methods and for establishing preferences among retrieved cases. A method for retrieving cases is using determinations [Russell]. A determination states that the solution feature of a problem is determined by another feature (e.g. nationality determines language).

```
(define (retrieve-by-pattern Retrieve-by-Protein)
    (current-case my-experiment)
    (pattern (define (experiment)
                (protein (>> protein current-case)))))
```

Domain knowledge can thus prescribe that the comparisons and preferences are made only to a subset of relevant cases. Next we have to compare the relevant retrieved cases in order to determine which is more relevant to our current problem.

4 Discussion

The notion of antiunification as generalization dates from 1970 [Plotkin]. There are some interesting articles on assessing similarity among structural representations. Bunke and Messmer [Bunke] define a distance between graphs $d(g, g')$ as the (minimum) cost sequence of edit operations that transforms g into g'.

A precedent of the approach taken here is the study on structural similarity of first order terms performed in [Jantke]. In this article, an algorithm for antiunifying first order terms is proposed. Part of the problem in doing so stems from the fact that it seems difficult to describe a case as one first order term: usually one would think of a set of first order terms as corresponding to a case.

The main goal of this paper has been to show that feature terms are a useful formalization that can be used to describe parts of the CBR process when using structured representations.

Acknowledgements

This work has been performed in the framework of Project analog, CICYT grant TIC-122/93. I am grateful to the discussions of the project members and to Josep-Lluís Arcos, Jaume Agustí, and Lluís Godo for enlightening debates on these and other topics.

References

[Aït-Kaci] H Aït-Kaci, A Podelski (1992), Towards a meaning of LIFE. PRL Research Report #11, Digital Research Laboratory.

[Arcos] Arcos, J. L., and Plaza, E. (1993), A Reflective Architecture for Integrated Memory-based Learning and Reasoning, In S. Wess, K.D. Althoff, M.M. Richter (Eds.), Topics in Case-Based Reasoning. Lecture Notes in Artificial Intelligence, 837, p. 289-300. Springer Verlag: Berlin.

[Armengol] Armengol, E. and Plaza, E. (1994b), Integrating Induction in a Case-based Reasoner. Proceedings of the Second European Workshop on Case-based Reasoning. Chantilly (France). pp. 243-251.

[Börner] Börner, K (1994), Structural similarity as a guidance in case-based design. In S Wess, K D Althoff, M M Richter (Eds.) Topics in Case-Based Reasoning, Lecture Notes in Artificial Intelligence, Vol. 837, p.197-208. Springer-Verlag.

[Bunke] Bunke, H and Messmer, B T (1994), Similarity measures for structured representations. In S Wess, K D Althoff, M M Richter (Eds.) Topics in Case-Based Reasoning, p. 106-118. Lecture Notes in Artificial Intelligence 837, Springer Verlag.

[Carpenter] Carpenter, B (1992), The Logic of Typed Feature Structures. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

[deRaedt] deRaedt, L (1992), Interactive Theory Revision. Academic Press: London.

[Jantke] K P Jantke (1993), Nonstandard concepts of similarity in case-based reasoning. Proceedings of the 17th Annual Conference of the "Gesellschaft für Klassifikation e.V.". Kaiserslautern, March 1993. Springer Verlag.

[Kettler] B P Kettler, J A Hendler, W A Anderson, M P Evett (1994), Massively parallel support for case-based planning. IEEE Expert, p. 8-14, Feb. 1994.

[López] B. López, E. Plaza (1991), Case-based Learning of Strategic Knowledge. Lecture Notes in Artificial Intelligence 482. Springer-Verlag 1991, pp. 398-411.

[Plaza] Plaza, E., Arcos, J.L. (1994), Integration of learning into a knowledge modelling framework. Lecture Notes in Artificial Intelligence, Vol. 867. Springer-Verlag 1994, pp. 355-373.

[Plotkin] Gordon D Plotkin, A note on inductive generalization. In B. Meltzer and D Michie (Eds.), Machine Intelligence 5, p. 153-163. Elsevier 1970.

[Russell] Russell, S. (1990), The Use of Knowledge in Analogy and Induction. Morgan Kaufmann.

Appendix A

In order to see that the antiunification results in a greatest lower bound (GLB) it suffices to see that for any lower bound of both F and F', then it is also a lower bound of $F \sqcap F'$ i.e. for any F^* s.t. $F^* \sqsubseteq F$ and $F^* \sqsubseteq F'$, then $F^* \sqsubseteq F_R$.

Recall the definition of subsumption as the h homomorphism in Definition 2.4. Given the hypothesis of F^*, there exist homomorphisms $h : Q^* \to Q$ and $h' : Q^* \to Q'$ such that the properties of Definition 2.4 hold. We will define an application $g : Q^* \to Q_R$ and show that g is an homomorphism as required by Definition 2.4 and thus $F^* \sqsubseteq F_R$.

Specifically we have to see that:

- $T^*(q) \sqsubseteq T_R(g(q))$
- $g(D^*(f, q)) = D_R(f, g(q))$

We will define g as follows: $g(q) = \{h(q), h'(q)\}$ then since $T(h(q)) \sqsupseteq T^*(q)$ and $T(h'(q)) \sqsupseteq T^*(q)$ it holds that $T^*(q) \sqsubseteq T(h(q)) \sqcap T(h'(q))$ for every q in Q^* and it follows from $T(h(q)) \sqcap T(h'(q)) = T_R(\{h(q), h'(q)\}) = T_R(g(q))$ that $T^*(q) \sqsubseteq T_R(g(q))$.

Moreover, for every q in Q^* and feature f such that $D^*(f, q)$ is defined it can be seen that $g(D^*(f, q)) = D_R(f, \{h(q), h'(q)\})$ since:

$$g(D^*(f,q)) = \{h(D^*(f,q)), h'(D^*(f,q))\} = \{D(f, h(q)), D(f, h'(q))\} = D_R(f, \{h(q), h'(q)\})$$