Adaptation with the INRECA - System

Wolfgang Wilke and Ralph Bergmann
Centre for Learning Systems and Applications
University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049, D-67653 Kaiserslautern, Germany
e-Mail: {wilke, bergmann}@informatik.uni-kl.de

1 Introduction

INRECA[1] offers tools and methods for developing, validating, and maintaining classification, diagnosis and decision support systems. INRECA's basic technologies are inductive and case-based reasoning [9]. INRECA fully integrates [2] both techniques within one environment and uses the respective advantages of both technologies. Its object-oriented representation language CASUEL [10, 3] allows the definition of complex case structures, relations, similarity measures, as well as background knowledge to be used for adaptation. The object-oriented representation language makes INRECA a domain independent tool for its destined kind of tasks.
When problems are solved via case-based reasoning, the primary kind of knowledge that is used during problem solving is the very specific knowledge contained in the cases. However, in many situations this specific knowledge by itself is not sufficient or appropriate to cope with all requirements of an application. Very often, background knowledge is available and/or necessary to better explore and interpret the available cases [1]. Such general knowledge may state dependencies between certain case features and can be used to infer additional, previously unknown features from the known ones. Furthermore, some applications require an adaptation of a retrieved case according to the actual problem at hand [13, 6, 4]. Therefore, general knowledge is required to specify such an adaptation. This paper addresses the representation and the processing of background knowledge required for case-based reasoning applications in the field of classification, diagnosis, and decision support with the INRECA-system. This work is part of the INRECA-project [9] and was driven by the needs of the developed applications.
In the INRECA-project we integrated the representation of background knowledge, represented as rules, with the object-oriented case representation. Moreover, it is crucial to avoid increased retrieval times as a consequence of a search-intensive inference procedure processing the knowledge. Otherwise, some of the advantages of case-based approaches compared to standard knowledge-based techniques would be lost. The background knowledge is not intended to be a substitution for the knowledge contained in the cases but a supplementation to the specific knowledge contained in the cases.

[1] This research was partially funded by the Commission of the European Communities (ESPRIT contract P6322, the INRECA project). The partners of INRECA are AcknoSoft (prime contractor, France), tecInno (Germany), Interactive Multimedia Systems (Ireland) and the University of Kaiserslautern (Germany) and partially funded by the Stiftung Innovation for Rheinland-Pfalz.

ECAI 96 Workshop: Adaptation in CBR.
Edited by A. Voss et al.
Published in 1996 by John Wiley & Sons, Ltd.

2 Representing and Using Background Knowledge in INRECA

In INRECA we use two different kind of rules to represent background knowledge:

- *Completion* rules to infer additional features from known features of an old case or the query. Thereby, these rules complete the description of a case.
- *Adaptation* rules to describe how an old case can be adapted to fit the current query.

In the following we will explain these two kinds of rules informally before going into the details of their representation.

2.1 Completion rules

In many situations, certain features of a case description directly depend on several other features. When the user enters some of the features in the query in, she/he should not be demanded to enter the values of features which are determined by the information she/he has already entered. Yet, simply ommiting these values in the query case leads to a less informed similarity assessment. Therefore, we introduce com-

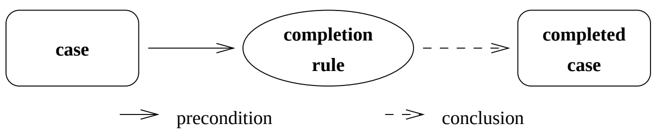

Figure 1. Completing case descriptions

pletion rules to extend the description of a case (see figure 1). The completion rules are an explicit modeling of attribute dependencies. These rules apply once to the cases of the case base as well as to the query case which is entered during consultation. After the adaptation process (see below), these rules are also applied to the attributes of the solution case.

Completion rules are used to infer values of attributes of the case description which are directly dependent on some other attributes of the case. Thereby, additional attributes can be assigned a value without asking the user. Furthermore, the occurrence of inconsistent values can be reduced. The attributes which are derived using the completion rules can then be used during the similarity assessment. Such a similarity assessment is based on the knowledge of more relevant attributes and should consequently be more precise. Since the completion rules are used to derive attributes of a case description which the user might also enter, the rules have to be correct in all situations. Uncertain, or just probable rules are not considered here. If a completion rule changes the value of an attribute which was previously provided by the user, this is considered as an error in the user's input.
Figure 1 describe the application of a rule to a case description. The rules are based on the values of attributes given in a specific case and as a result of their application the rules add certain attribute values to this case.

2.1.1 Informal Example

As a first example for using completion rules we want to introduce an application in a travel agency domain[8] in which several descriptions of available journeys are collected in a case base. Each journey is described by several attributes such as: *destination, duration, price, kind of transportation, kind of accommodation*, etc.

Assume a case representation for a journey which includes the specification of the number of adults and the number of children which are involved in the journey. Moreover, assume that the representation also specifies the total number of persons because for several journeys only the total number of people is important (e.g. in an apartment). In this situation, the following general rule is useful:

> *The total number of persons is always the sum of the number of children and the number of adults.*

This rule reliefs the user from the necessity to enter the value for the total number of persons when the number of children and the number of adults already have been entered.

2.2 Adaptation Rules

Adaptation rules come into play after a case has been retrieved [1]. Most likely, this case does not completely fit the requirements specified in the user's query:

Some attributes of the retrieved case might differ somehow while others may exactly match the query. According to the differences, the retrieved case must be modified to become better suited to the current query. As shown in figure 2, the precondition of adaptation rules can refer to attributes of the retrieved case, attributes of the current query case, and already established attributes of the solution case in the precondition of the rule. In a rule's conclusion, new attribute values for the solution case can be specified.

2.2.1 Informal example

Suppose, the user's query specifies a journey with a duration of two weeks. Furthermore, assume that the most similar case

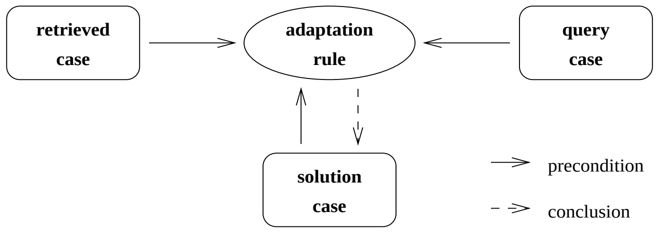

Figure 2. Cases used for adaptation

is a journey which usually takes one week. Since the price for this journey is calculated on a one week basis, it must be adapted to correctly refer to the two week journey as specified in the query:

> *If the duration specified in the query case is longer than the duration specified in the retrieved case, then the price in the solution case is computed by adding the price of the retrieved case and the price for accommodation for the additional period of time.*

2.3 Architecture for Processing Background Knowledge

Figure 3 shows how the general knowledge is processed within the INRECA-system. Two components are used for processing the rules: one component for case completion and one component for case adaptation. The case completion component applies the completion rules to extend the representation of the available cases before they are stored in the case base. The same component can also be used to complete the query presented by the user of the system. The case retrieval component uses the completed case descriptions only. Arbitrary retrieval approaches can be used for this task. Finally, the case adaptation component applies the adaptation rules to compute a solution case out of the retrieved case and the query case. This solution case can then be stored again in the case base for future use.

2.4 Impact of the Object-Oriented Case Representation

Obviously, object-oriented case representation has a strong impact on the mechanisms which handle the completion and adaptation rules. Given this kind of case representation, we have identified the classes to be the most natural place to attach the rules to. Within the scope of a class, a rule has access to the slots which are defined for an instance of that class, including the instance slots which are inherited from its superclasses. Additionally, rules must be given access to slots of those objects which are related to instances of the class the rule belongs to. In the same manner as slots are inherited from the superclass to a class, the rules can also be inherited. Rules which are defined for a class are also valid for all objects subclasses. Figure 4 shows an example of the simultaneous occurrence of inherited slots and related objects. Additionally, the figure shows different sets of rules which are attached to the classes and indicates the slots to which these

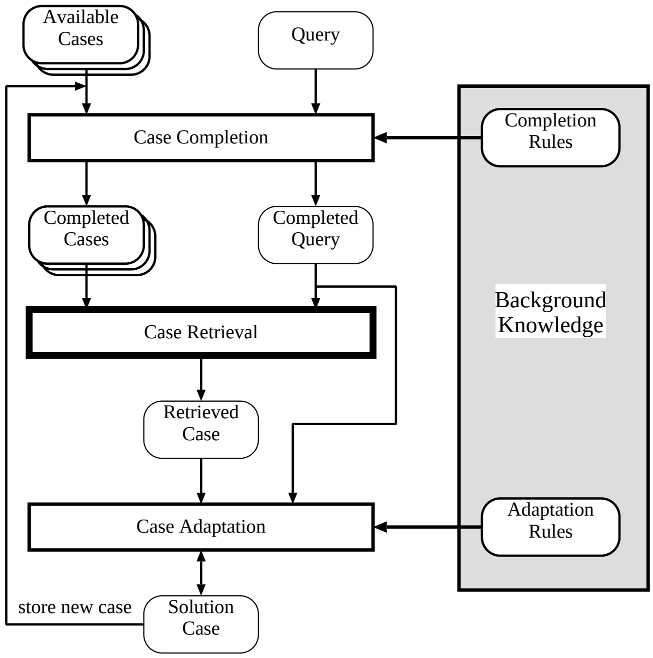

Figure 3. Architecture for integrating background knowledge into case-based reasoning in INRECA

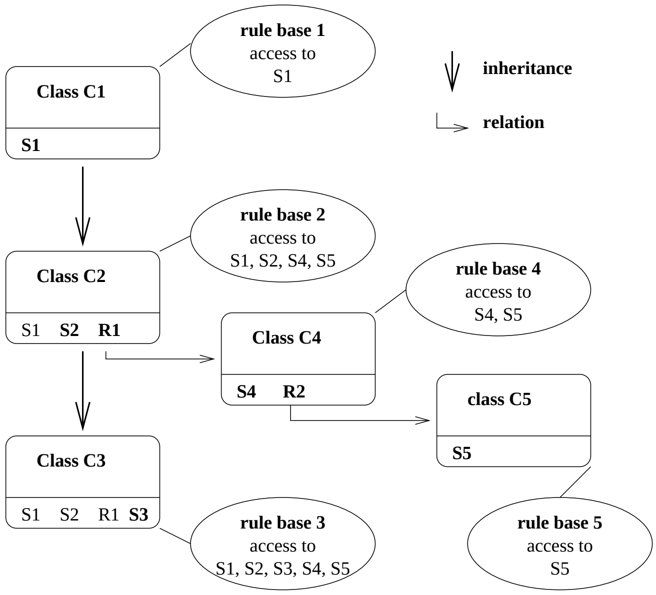

Figure 4. Scope of rules in the object-oriented case representation

rules have access to. The figure shows five different classes C1, . . . , C5 where C2 is a direct subclass of C1 and C3 is a direct subclass of C2. Each class defines one non-relational slot, i.e. a slot which can hold values of basic types, but not objects. These slots are named S1, . . . , S5, respectively. Moreover, class C2 and class C4 define relational slots R1 and R2, respectively. To illustrate the scope of the rules associated with the five classes, the instance slots that can be accessed by each of the rules are shown. For example, we can see that rules of rule base 2 have access to the instance slots of their own class (S2), to the slots of their superclasses (S1), and to the slots which are available in related classes (S4, S5). To make a precise reference to instance slots of related classes, the relation itself (e.g. R1) must always be noted together with the respective slot (a possible notation would be: R1→S4 or R1→R2→S5). Due to the inheritance of the rules, the rules of rule base 1 are also valid for all instances of the classes C2 and C3, but of course not for instances of the classes C4 or C5, since class C1 is not a superclass of C4 or C5. Applying the object-oriented representation also to rules provides an efficient way of expressing background knowledge. Due to the rule inheritance, knowledge which applies to many different objects can be expressed in rules which are attached to an appropiate superclass these objects are a kind of. Moreover, the restricted set of slots a rule can access still maintains the principle of information encapsulation of object-oriented representations.

2.4.1 Completion Rules

Each rule consists of two parts: a precondition part and a conclusion part. The precondition part defines a conjunction of conditions. Each condition must be expressed in terms of the accessible slots with respect to the class the rule belongs to. A condition can compare the value of a slot to values of other slots, constants, or local variables. Moreover, the precondition include the computation of values of local variables from other values by arbitrary functions. The conclusion part of a rule consists of a set of actions which are executed if the precondition is valid, i.e. all conditions in the precondition are fulfilled. An action in the conclusion of a rule can assign a value to a slot, create a new object for a relational slot or specialize the class of an already existing object in a relational slot.

2.4.2 Adaptation Rules

The basic difference between completion rules and adaptation rules is that completion rules only refer to one case, while adaptation rules always refer to three cases, namely the query case, the retrieved case, and the solution case (see figure 2). These three different cases might be taken into account when specifying the precondition and the conclusion of adaptation rules. The precondition of an adaptation rule may consist of the same elements as the preconditions of a completion rule. However, since an adaptation rule might take into account three different cases, each reference to a slot must also state to which of the three possible cases the slot belongs to. The conclusion of an adaptation rule may consist of the same functions as the actions of a completion rule. Contrary to the precondition part, it is not necessary to explicit specify the case when an assignment to a slot is made in the conclusion part, because the solution case is the only case that may be modified by an adaptation rule.

2.5 An Example

We now want to give an example of a completion rule and an adaptation rule. For that purpose we represent the rules from the travel agency domain introduced in [8]. We assume that the descriptive model contains three classes: the classes *vacation*, *transportation* and *accommodation*. In our model, the classes *transportation* and *accommodation* describe objects which are directly related to the *vacation* object as shown in figure 5. This figure also shows some slots needed to describe the objects. Based on this model, the following comple-

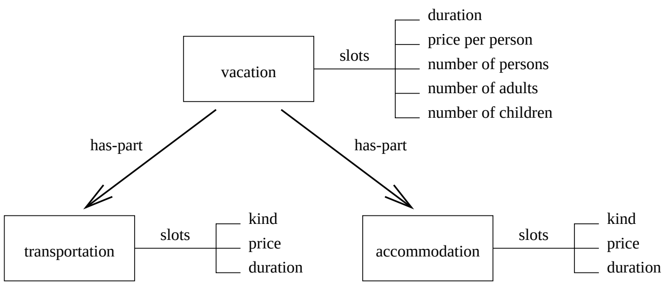

Figure 5. Descriptive Model of the Travel Agency Domain

tion rule can be formulated to express that *the total number of persons is always the sum of the number of children and the number of adults.*

```
defrule person_calculation of class vacation
   rule
      ?x := number_of_adults + number_of_children
   ⟹
      number_of_persons := ?x.
```

This rule computes the sum of the values in the two slots *number_of_adults* and *number_of_children* and assigns the result to the variable named ?x. In the conclusion of the rule, the slot *number_of_persons* is assigned the value stored in ?x. The adaptation rule for adapting the price of a journey with respect to its duration can be formulated as follows:

```
defadaptationrule price_adaptation of class vacation
 rule
    query duration > retrieved duration &
    ?additional_days := query duration −
                  retrieved duration &
    ?additional_price := ?additional_days ∗
                  retrieved accommodation→price &
    ?new_price := retrieved price + ?additional_price
 ⟹
    solution price := ?new_price.
```

This adaptation rule first tests whether the duration in the query case is longer than the duration in the retrieved case. Then the difference between the required and retrieved duration is computed and the additional price is determined. Finally, the last calculation adds the additional price to the price already stated in the retrieved case. In the conclusion of the adaptation rule, the price slot of the solution case is assigned the new price.

2.6 Processing Rules

Controlling the completion and adaptation rules can be realized a forward-chaining rule interpreter with an underlying Rete-Network [7]. Due to the existence of *a-kind-of* and *has-part* relations in an object-oriented representation, one has to be prepared for a great challenge if one tries to use one of the standard “flat” forward chaining rule systems. One first approach towards the integration of object-oriented data structures and a forward-chaining rule interpreter is the NéOpus system [11]. This system offers a mechanism for structuring large sets of rules in separate *rule-bases*. It is also possible to define subbases of rule-bases, where a subbase inherits the rules of its superbase. Because of these features, NéOpus was chosen as the rule engine for the INRECA-system. The completion and adaptation rules, respectively, should be grouped into distinct rule-bases belonging to the different classes of the descriptive model. Every rule-base may have its own Rete-Network. The nodes of such a network are the single premises and conclusions of the rules of the associated rule-base. The edges of such a directed network connect the premises and conclusion that belong to a single rule in such a way that if one starts at the first premise of a rule and follows the edges along a path to a leave node, one ends up at the node representing the conclusion of the rule. If a new case is added to the system, it is propagated through the network while every node tests whether the premise it represents holds for the given case. The case is passed on to the next node only if the test succeeds. If finally a rete node representing the conclusion of a rule is reached, the rule fires, i.e. the actions of the conclusion are executed. As these actions modify certain slots of the case, it is important to propagate these changes through the Rete-Network, and since there is one network per class, it is necessary to propagate the change of an attribute through all existing networks. In order to handle this propagation in an efficient way, it is advantageous to store a global table in which every attribute (qualified by the class the attribute belongs to) is associated with the Rete nodes that are affected by a change of the attribute's value. The performance of such an architecture is similar to the performance of a single “flat” forward-chaining rule system that uses a Rete-Network with the cumulative size of the suggested distinct networks.

3 Appendix

This section addresses the specific questions to be discussed during the workshop.

3.1 Adaptation Knowledge

For adaptation, INRECA requires a set of adaptation rules. The primary adaptation knowledge is coded in the adaptation rules. They infer a solution case from the query case and a similar retrieved case. The completion rules represent another kind of general knowledge. They complete the query case during consultation, the retrieved cases in the case base and especially the solution case during the adaptation process with additional attributes derived from existing ones. The paper described the organisation of the rules in detail.

3.2 Case structure

INRECA is a case-based reasoning shell for solving analytic tasks. If you solve a decision support problem with INRECA it is an analytic one. However, there are also decision support problems that are, or involve syntetic tasks.
In INRECA we use an object-oriented case representation, where an object contains a set of attribute-value pairs. These attributes describe the characteristic features of an object. So, the case description is structured, but not into real subcases with CBR mechanism of its own. There is the possibility to distinguish explicitly between a problem and a solution part in the case description but this is not required. Attributes and objects are divided into discriminant and non-discriminant ones in the domain model. As a consequence, the calculated similarity between cases depend only on discriminant attributes and objects. Also the indexing mechanism, the *kd-tree* [15, 16, 14], prefers only discriminant attributes for an effective indexing of the case base. But an indexing on all attributes is possible, also on those describing the solution. For adaptation, discriminant as well as non-discriminant attributes and objects are taken into account. There is no restriction to mark attributes and objects as discriminant or not.

3.3 Retrieval and Indexing

The retrieval calculates a weighted sum of selected attribute similarities to find the most similar one. The similarity measure can be specified by the user. This does not guarantee that the selected case is adaptable. So, the similarity measure is responsible to ensure that the retrieved case together with the query case produce a proper solution after the adaptation process. INRECA offers different learning algorithms optimize the indexing of the case-base[14] in order to improve the correctness of the system. There is also the possibility to exclude cases form the case-base before the retrieval starts, with a set of exclusion rules given by the user. In both cases, background knowledge is used, but there is no explicit use of the defined adaptation rules or completion rules.

3.4 Adaptation

INRECA is a domain independent system. So it is difficult to answer these questions in general. In simple classification tasks, usually only the target slot of the retrieved case is copied, perhaps together with some additional information about the desired class. So, there is no need for adaptation. In synthetic tasks like decision support, adaptation is more relevant to solve the problem. In general, adaptation in INRECA is necessary because otherwise a very large case-base would be required to tackle real-life problems. The retrieved cases are not combined during the adaptation process. Adaptation depends only on the query case and a similar retrieved case. This selection is one co-operation with the user during adaptation. Furthermore, there is the possibility to check the consistency of the adapted solution with the completion rules.

3.5 Strength, Limitations and Evaluation

In this paper we presented the INRECA approach to integrating general knowledge represented as rules with object-oriented case representation and reasoning. Rules, which are attached to the classes, have a clearly defined scope of slots which they can access and modify, and rules can be inherited by subclasses. With the chosen uniform representation of general knowledge, case descriptions can be extended to achieve a more informed retrieval of cases and a transformational adaptation approach [5, 6] can be realized. This object-oriented rule mechanism allows to efficiently represent and process general knowledge for a particular class of case-based reasoning applications. However, the user is responsible for the correctness of the rules and for defining similarity measures that allow to retrieve an adaptable case [12]. Currently, our approach does not include mechanisms to automatically retrieve cases for which it is known in advance that they can be adapted by the available adaptation rules. Moreover, our approach is limited to transformational adaptation, which is sufficient for most classification and diagnosis applications. Currently, neither knowledge required to perform derivational adaptation [5, 13] (as required for planning tasks) nor knowledge which is naturally expressed as a set of constraints (as required for design task) can be represented and processed within our approach.

Acknowledgements

The authors want to thank Ivo Vollrath for the realization of this work and Harald Holz and Ivo Vollrath for the helpful discussions to improve this paper.

REFERENCES

[1] A. Aamodt and E. Plaza, 'Case-based reasoning: Foundational issues, methodological variations, and system approaches', *AI Communications*, 39–59, (1994).

[2] Eric Auriol, Michel Manago, Klaus-Dieter Althoff, Stefan Wess, and Stefan Dittrich, 'Integrating induction and case-based reasoning: Methodological appraoch and first evaluations', in *Proceedings EWCBR-94*, (1994).

[3] Ralph Bergmann. CASUEL: A Common Case Representation Language. URL: http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/, 1994.

[4] Ralph Bergmann, Gerd Pews, and Wolfgang Wilke, 'Explanation-based similarity: A unifiying approach for integrating domain knowledge into case-based reasoning for diagnosis and planning tasks', in *Topics in Case-Based Reasoning*, eds., Stefan Wess, Klaus-Dieter Althoff, and Michael M. Richter, number 837 in LNAI, 182–196, Springer Verlag, (1994).

[5] J.G. Carbonell, 'Derivational analogy : A theory of reconstructive problem solving and expertise acquisition', in *Machine Learning : An Artificial Intelligence Approach, Vol II*, eds., R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Morgan Kaufmann Publishers, (1986).

[6] P. Cunningham, D. Finn, and S. Slattery, 'Knowledge engineering requirements in derivational analogy', in *Topics in Case-Based Reasoning*, eds., S. Wess, A. Althoff, and M.M. Richter, Lecture Notes in Artificial Intelligence, pp. 234–245. Springer Verlag, (1994).

[7] C.L. Forgy, 'A fast algorithm for many pattern/many object pattern match problem', *Artificial Inteligence*, **19**, 17–37, (1982).

[8] Mario Lenz, 'Cabata - a hybrid cbr system', in *Proceedings First European Workshop on Case-based Reasoning*, eds., M.M. Richter, S. Wess, K.D. Althoff, and F. Maurer, volume I, pp. 204–209, (1993).

[9] M. Manago, K.D. Althoff, E. Auriol, R. Traphöner, S. Wess, N. Conruyt, and F. Maurer, 'Induction and reasoning from cases', in *First European Workshop on Case-Based Reasoning (EWCBR-93)*, eds., M.M. Richter, S. Wess, K.D. Althoff, and F. Maurer, pp. 3313–318, (1993).

[10] Michel Manago, Ralph Bergmann, Stefan Wess, and Ralph Traphöner, 'CASUEL: A common case representation language - version 2.0', Technical report, Esprit Project INRECA Deliverable D1, (1994).

[11] Francois Pachet, 'Reasoning with objects: The neopus environment', Technical Report 13/91, University of Paris VI, (1991).

[12] B. Smyth and M.T. Keane, 'Retrieving adaptable cases', in *Topics in Case-Based Reasoning*, eds., S. Wess, A. Althoff, and M.M. Richter, Lecture Notes in Artificial Intelligence. Springer Verlag, (1994).

[13] M. M. Veloso, *Learning by analogical reasoning in general problem solving*, Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 1992.

[14] S. Wess, *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*, Ph.D. dissertation, University of Kaiserslautern, 1995.

[15] Stefan Wess, Klaus-Dieter Althoff, and Guido Derwand, 'Improving the retrieval step in case-based reasoning', in *Proceedings First European Workshop on Case-Based Reasoning*, pp. 83–88, (1993).

[16] Stefan Wess, Klaus-Dieter Althoff, and Guido Derwand, 'Using kd-trees to improve the retrieval step in case-based reasoning', in *Topics in Case-Based Reasoning*, pp. 83–88, (1994).