Integrating General Knowledge with Object-Oriented Case Representation and Reasoning

Ralph Bergmann[1], Wolfgang Wilke[1], Ivo Vollrath[1], Stefan Wess[2]

[1]Centre for Learning Systems and Applications,
University of Kaiserslautern,
P.O. Box 3049,
D-67653 Kaiserslautern,
Germany

[2]Inference GmbH,
Lise-Meitner-Strasse,
D-85716 Unterschleissheim,
Germany

Abstract

When problems are solved through reasoning from cases, the primary kind of knowledge is contained in the specific cases which are stored in the case base. However, in many situations additional background-knowledge is required to cope with the requirements of an application. We describe an approach to integrate such general knowledge into the reasoning process in a way that it complements the knowledge contained in the cases. This general knowledge itself is not sufficient to perform any kind of model-based problem solving, but it is required to interpret the available cases appropriately. Background knowledge is expressed by two different kinds of rules that both must be formalized by the knowledge engineer: *Completion* rules describe how to infer additional features out of known features of an old case or the current query case. *Adaptation* rules describe how an old case can be adapted to fit the current query. This paper shows how these kinds of rules can be integrated into an object-oriented case representation.

1 Introduction

When problems are solved through 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, general knowledge is available and necessary to better explore and interpret the available cases [Aamodt and Plaza, 1994]. Such general knowledge may state dependencies of certain features of a case 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 [Veloso, 1992; Cunningham *et al.*, 1994; Bergmann *et al.*, 1994]. Therefore, general knowledge is required to specify such an adaptation. Adaptation abilities are much more essential for synthetic tasks such as design or planning but several applications from the field of classification, diagnosis, or decision support also require at least some simple adaptation capabilities, typically in a transformational style. This paper addresses the representation and the processing of general knowledge (sometimes called background knowledge) required for case-based reasoning applications in the field of classification, diagnosis, and decision support. This work is part of the INRECA-project [Manago *et al.*, 1993] and was driven by the needs of the applications developed as part of the project. From a knowledge engineering perspective the representation of general knowledge should be tightly coupled with the mechanisms used for case representation. Today, object-oriented case representations such as CASUEL [Manago *et al.*, 1994] have proven to be flexible and efficient and allow to naturally model the complexities of real cases. Therefore, we want to integrate the representation of general knowledge, represented as rules, with such an 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 model-based techniques will be lost. In particular, we want to point out that the aim of this paper is not an integration of different reasoning paradigms such as model-based and case-based reasoning (e.g. [Aamodt, 1991; Voss *et al.*, 1993; Bergmann *et al.*, 1994]). The general knowledge is not intended to be a substitution for the knowledge contained in the cases but an addition to the specific knowledge of the cases.

2 Representing and Using Background Knowledge

2.1 Kinds of Rules

We have identified two kinds of rules to be essential:

- *Completion* rules infer additional features out of known features of an old case or the query. Thereby, these rules complete the description of a case.
- *Adaptation* rules 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.

2.1.1 Completion Rules

In many situations, certain features of a case description are directly dependent on several other features. When the user enters some of the features in the query, she/he should generally not be demanded to enter the values of features which are absolutely determined by the information she/he has already entered. But not having these values in the query case leads to a less informed similarity assessment. Therefore, we propose completion rules to extend the description of a case (see figure 1). These rules apply to the cases of the case base as well as to the query case which is entered during consultation. 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 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 must be known to be true 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 shows these preconditions and conclusions of

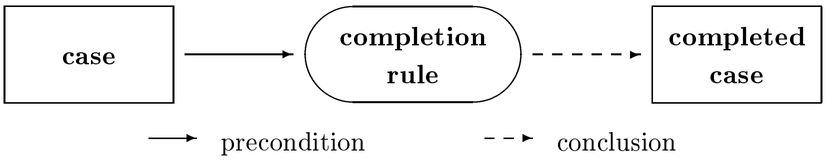

Figure 1: Completing case descriptions

a completion rule. The rule is based on the values of attributes given in a specific case and as a result of it's application the rule may add certain attribute values to this case.

2.1.2 Adaptation Rules

Adaptation rules come into play after a case is retrieved [Aamodt and Plaza, 1994]. Most likely, this case does not fully fit the requirements of the user. Some attributes of the retrieved case may exactly match the query while others might differ somehow. According to these differences, the retrieved case must be modified to become better suited to the current query. As shown in figure 2, adaptation rules combine

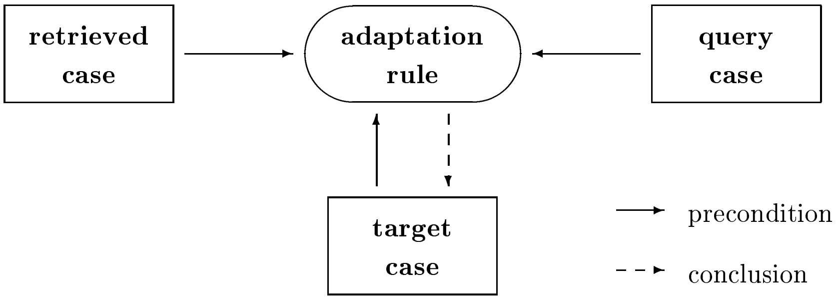

Figure 2: Cases used for adaptation

attributes of the retrieved case, attributes of the current query case, and already derived attributes of the target case in the precondition of the rule. In a rule's conclusion, new attribute values for the target case are derived.

2.2 Architecture for Processing General Knowledge

Figure 3 presents an architecture that shows how the general knowledge can be processed within a case-based reasoning system. Two additional compontents are required for processing the rules: one

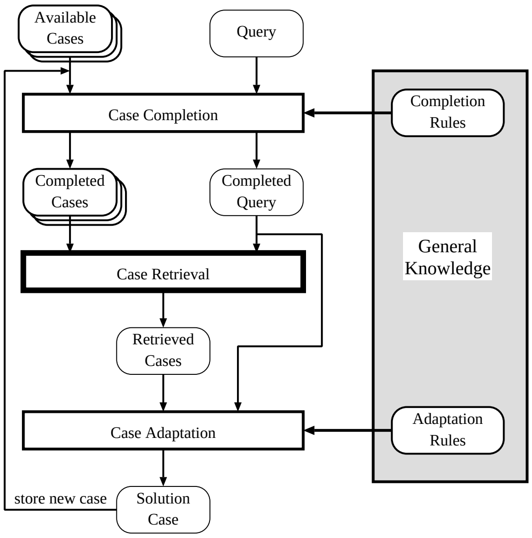

Figure 3: Architecture for integrating general knowledge into case-based reasoning

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 works on the completed cases only. Arbitrary retrieval approaches can be used for this task. In INRECA, all four different levels of integrating induction and case-based reasoning [Manago *et al.*, 1993] can be used for case retrieval. 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.

2.3 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 direct access to the slots which are defined for that class and to those slots which are inherited from its superclasses. Additionally, rules must be given access to slots of those objects which are related to the object 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 superclass are always valid for all subclasses. Figure

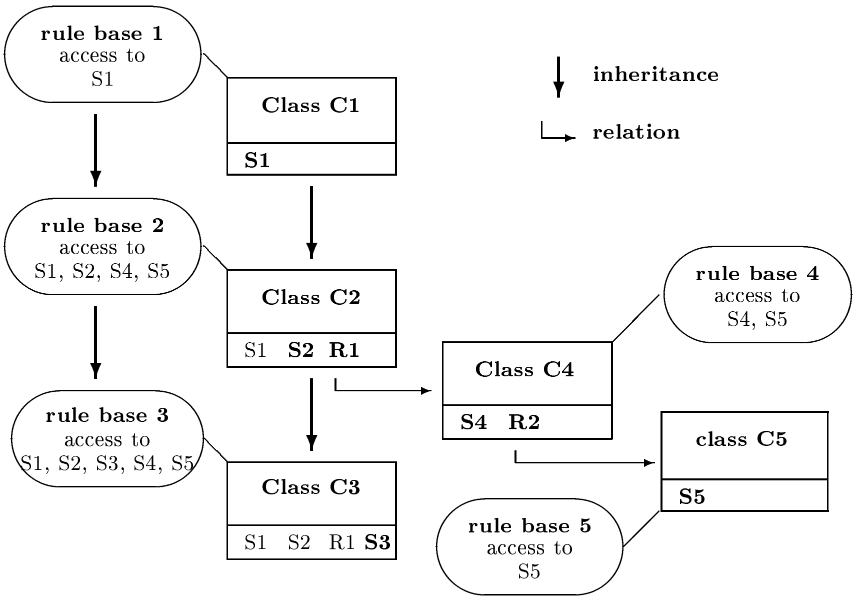

Figure 4: Scope of rules in the object-oriented case representation

4 shows an example of the simultaneous occurrence of inherited and related objects. Additionally, the figure shows different sets of rules which are attached to the classes and indicates the slots to which these rules have access to. The figure shows five different classes C1, ..., C5 where C2 is a subclass of C1 and C3 is a subclass of C2. Each class has one none-relational slot, i.e. slots which can hold values of basic types, but not objects. These slots are named S1, ..., S5 respectively. Moreover, class C2 and class C4 have relational slots R1 and R2, respectively. To illustrate the scope of the rules associated with the five classes, the 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 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 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 objects of the classes C2 and C3, but of course not for objects of the classes C4 and C5 since class C1 is not a superclass of C4 and C5. Applying the object-oriented representation also to rules enables 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 the respective superclass these objects belong to. Moreover, the restricted set of slots a rule can access still maintains the principle of information encapsulation of object-oriented representations.

2.3.1 Completion Rules

Each completion 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 to which 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 can also be used to specify an arbitrary function which calculates a new value using the existing slot values. The conclusion part of a rule consists of a set of actions which are executed if the precondition is fulfilled, 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.

2.3.2 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 target case (see figure 2). These three different cases have to be taken into account when specifying the preconditions and the conclusion of adaptation rules. The preconditions of an adaptation rule may consist of the same elements as the preconditions of a completion rule, but because an adaptation rule has to take into account three different cases as explained earlier, each reference to a slot must also state which of the three possible cases is meant to be referenced. The conclusion of an adaptation rule may consist of the same elements as the actions of a completion rule. Contrary to the precondition part, it is not neccessary to explicitly state the case when an assignment to a slot is made in the conclusion part because the target case is the only case that may be modified by an adaptation rule.

2.4 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 [Lenz, 1993]. 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 completion 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 adds the values of 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:

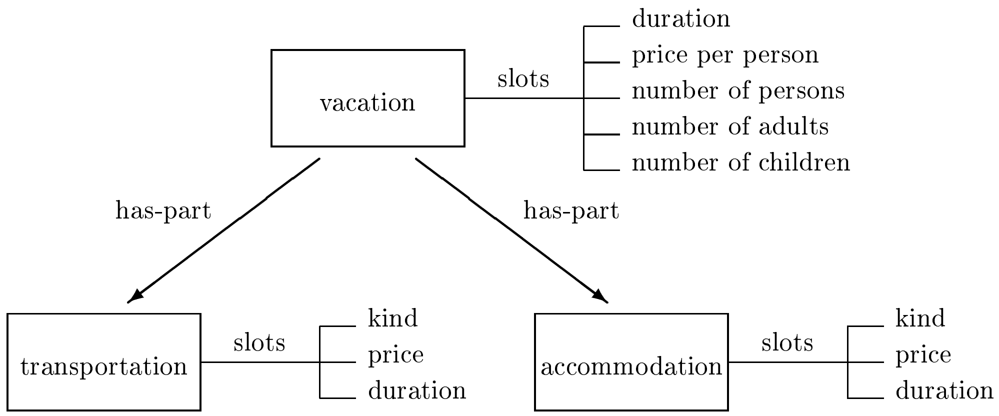

Figure 5: Descriptive Model of the Travel Agency Domain

```
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
   =⇒
     target 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 target case is assigned the new price.

2.5 Processing Rules

Controlling the completion and adaptation rules can be realized by some kind of forward chaining rule interpreter with an underlying Rete-Network [Forgy, 1982]. Due to the existance of *a-kind-of* and *has-part* relations in an object-oriented representation, one has to be prepared for a great challenge if he 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 [Pachet, 1991]. This system offers a mechanism for structuring large sets of rules in seperate *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 rulebases belonging to the different classes of the descriptive model. Every rule base may have it's own Rete-Network. The nodes of such a network are the single premisses and conclusions of the rules of the associated rule base. The edges of such a directed network connect the premisses and conclusion that belong to a single rule in such a way that if one starts at the first premisse of a rule and searches his way along the path, he 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 premisse 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 and 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 concept class, it is neccessary 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 Discussion

In this paper we presented a new 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 developed uniform representation of general knowledge, case descriptions can be extended to achieve a more informed retrieval of cases and a transformational adaptation approach [Carbonell, 1986; Cunningham *et al.*, 1994] can be realized. This object-oriented rule mechanism allows to efficently represent and process general knowledge for a particular class of case-based reasoning applications. However, the user is fully responsible for the correctness of the rules and for defining similarity measures that allow to retrieve an adaptable case [Smyth and Keane, 1994]. 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, the knowledge required to perform derivational adaptation [Carbonell, 1986; Veloso, 1992] as required for planning tasks or knowledge which is naturally expressed as a set of constraints as required for design task cannot be represented and processed within our approach.

Acknowledgement This work was funded by the INRECA project (ESPRIT-contract P6322). The partners of INRECA are AcknoSoft (prime contractor, France), tecInno GmbH (Kaiserslautern, Germany), IMS (Dublin, Ireland) and the University of Kaiserslautern (Germany).

References

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

[Aamodt, 1991] Agnar Aamodt. *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning*. PhD thesis, University of Trondheim, 1991.

[Althoff *et al.*, 1995] K.-D. Althoff, E. Auriol, R. Bergmann, S. Breen, S. Dittrich, R. Johnston, M. Manago, R. Traphöner, and S. Wess. Case-based reasoning for decision support and diagnostic problem solving: The inreca approach. In B. Bartsch-Spörl, D. Janetzko, and S. Wess, editors, *Fallbasiertes Schließen – Grundlagen & Anwendungen: Workshop auf der 3. Deutschen Expertensystemtagung (XPS-95)*. Centre for Learning Systems and Applications, University of Kaiserslautern, 1995.

[Bergmann *et al.*, 1994] R. Bergmann, G. Pews, and W. Wilke. Explanation-based similarity: A unifying approach for integrating domain knowledge into case-based reasoning. In M.M. Richter, S. Wess, K.D. Althoff, and F. Maurer, editors, *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes on Artificial Intelligence*, pages 182–196. Springer, 1994.

[Booch, 1991] Graddy Booch. *Object-Oriented Design with Applications*. Benjamin/Cummings, 1991.

[Carbonell, 1986] J.G. Carbonell. Derivational analogy : A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : An Artificial Intelligence Approach, Vol II.* Morgan Kaufmann Publishers, 1986.

[Cunis, 1993] Roman Cunis. *Das 3-stufige Frame-Repräsentationsschema - eine mehrdimensional modulare Basis für die Entwicklung von Expertensystemkernen.* PhD thesis, Universität Hamburg, Fachbereich Informatik, 1993.

[Cunningham *et al.*, 1994] P. Cunningham, D. Finn, and S. Slattery. Knowledge engineering requirements in derivational analogy. In S. Wess, A. Althoff, and M.M. Richter, editors, *Topics in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, pages 234–245. Springer Verlag, 1994.

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

[Lenz, 1993] Mario Lenz. Cabata - a hybrid cbr system. In M.M. Richter, S. Wess, K.D. Althoff, and F. Maurer, editors, *Proceedings First European Workshop on Case-based Reasoning*, volume I, pages 204–209, 1993.

[Manago and Auriol, 1995] M. Manago and E. Auriol. Integrating induction and case-based reasoning for troubleshooting cfm-56 aircraft engines. In B. Bartsch-Spörl, D. Janetzko, and S. Wess, editors, *Fallbasiertes Schließen - Grundlagen und Anwendungen*, number LSA-95-02, pages 73–80, Centre for Learning Systems and Applications, University of Kaiserslautern, 1995.

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

[Manago *et al.*, 1994] 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.

[Minsky, 1975] Marvin Minsky. A framework for representing knowledge. In P.H. Winston, editor, *The psychology of computer vision*, pages 211–277. McGraw-Hill, 1975.

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

[Smyth and Keane, 1994] B. Smyth and M.T. Keane. Retrieving adaptable cases. In S. Wess, A. Althoff, and M.M. Richter, editors, *Topics in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence. Springer Verlag, 1994.

[Taylor, 1992] David A. Taylor. *Object-Oriented Information Systems: Planning and Implementation.* John Wiley, 1992.

[Tversky, 1977] A. Tversky. Features of similarity. *Psychological Review 84*, pages 327 – 352, 1977.

[Veloso, 1992] M. M. Veloso. *Learning by analogical reasoning in general problem solving.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1992.

[Voss *et al.*, 1993] A. Voss, B. Bartsch-Spörl, L. Hovestadt, K.P. Jantke, U. Peterson, and G. Strube. Fabel: Projektstatus, perspektiven und potentiale. Technical Report Fabel Report No. 16, GMD, Sankt Augustin, Bonn, Germany, 1993.

[Wess, 1995] S. Wess. *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik.* PhD thesis, University of Kaiserslautern, 1995.