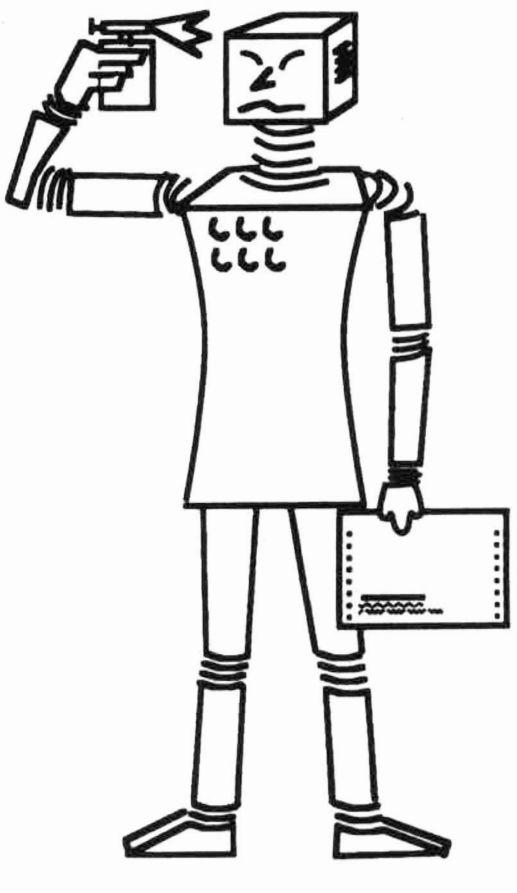
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT



Retrieval in Case-Based Reasoning using preferred subtheories

Michael Mehl

SEKI Report SR-92-14 (SFB)

Retrieval in Case-Based Reasoning using preferred subtheories

Michael Mehl
Universität Kaiserslautern*, Fachbereich Informatik
Arbeitsgruppe Prof. Dr. Michael M. Richter

now
DFKI
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3
D-W 6600 Saarbrücken 11
e-mail: mehl@dfki.uni-sb.de

Abstract

The goal of this work is to develop a formal logical foundation of the representation and the retrieval of cases in case–based reasoning. An adequate basis therefor provides the default logic with priorities. We present transformations which construct defaults from the memory of cases such that the retrieval of knowledge in case–based reasoning corresponds roughly to the preferred subtheory obtained by the defaults.

1 Introduction

Case–Based Reasoning(CBR) is one of the current issues in expert system research. An overview is given in [RisslandKolodnerWaltz89] and [Slade91]. The underlying principle of CBR is to memorize cases and organize them efficiently to retrieve knowledge when new problems arise. These information guide solving similar problems adopting the recorded known solutions. CBR consists of three main tasks: (1) organizing the memory structure, (2) retrieving information from memory and (3) using this information for problem solving. Our paper is focused on the second step, the information retrieval task.

A commonly used model for an efficient organization of cases in CBR is the dynamic memory approach developed by [Schank82]. This approach was further sophisticated by [Kolodner83a, Kolodner83b, Kolodner84]. It will be the basis for the following sections. Cases are inserted in a hierarchical graph structure, called case memory. The nodes of the graph contain generalized information about the subsumed cases and the links between the nodes control the retrieval of information.

The problem of such a generalization is the correctness: in order to achieve an efficient and compact representation it is necessary to subsume many cases under one generalization rather than considering every special case. The effect is that increasing knowledge about new cases during the retrieval process does not imply monotonic growing of information derived from the case memory.

*The work presented herein was partially supported by "Deutsche Forschungsgemeinschaft (DFG)", Sonderforschungsbereich 314: "Artificial Intelligence - Knowledge-Based Systems", project X9. It was developed at the university of Kaiserslautern.

Because the hierarchical structure of the case memory contains knowledge about the preferences of information in nodes it is desirable to argue about preferences among formulas in the logic. A further requirement is that the logic is well understood and easy to apply. Both wishes are fulfilled by the default logic [Reiter80] and the extension to prioritized defaults [Brewka89]. The main advantage of prioritized defaults is their possibility to solve conflicts in the case of competing defaults and to reduce the number of extensions. Extensions are the possible theories supported by the given defaults.

2 Formal framework

Now a formal definition of the case memory and the prioritized default theory is presented to lay the foundations for the combination of both.

2.1 The case memory

The case memory provides an efficient model for the storage and organization of cases for case-based reasoning. The main goal is to extract knowledge from a case base that guides the solution of new problems represented also by (possibly incompletely specified) cases. The case memory is a self organizing memory in the sense of dynamically reorganizing its structure if new cases are inserted. This adaptation process is not the topic of this work. Hereinafter an already constructed, complete case memory constitutes the basis for further examination of the retrieval process.

Cases are described by features and in general contain further information, especially solutions for problems, e.g. in diagnosis the cases are described by observed symptoms and contain the reasons for the recognized errors. The meaning of cases herein is more general because no differences between the features and the additional information associated with a case are made. All the information about the case is thus represented in an uniform way.

The simple model for a case is that it is a set of attribute–value pairs, called features. Any subset can perhaps (definitely) characterize a case.

Definition 2.1 (Case)
Let A be a finite set of attributes, V a finite set of values.
$F \subseteq A \times V$ is a case *iff* $\forall a \in A \,.\, \forall v, v' \in V$
$(a, v) \in F \land (a, v') \in F \Rightarrow v = v'$ *(F can be seen as a partial function)*
A set of cases $\mathcal{F}$ is called a case base.

This definition means for a case F: if $(a, v) \in F$, then F has the value v for the attribute a. If the set of attributes A contains an a such that $\forall v \in V \,.\, (a, v) \notin F$, then the attribute a is undefined, unknown or irrelevant for the case F.

In the case memory the cases with common features are organized together in so called generalizations or GENS. GENS include attribute–value pairs, so called NORMS, which describe features for the most cases subsumed by the GEN. In GENS the cases are indexed by discriminating features, called DIFFS. These DIFFS index not only the cases beneath a GEN, but also more specialized GENS. The case memory represents an acyclic directed graph, in which the cases are the leafs and GENS are inner nodes. The connections between nodes are marked with the DIFFS.

Before defining the case memory we need some functions for the exploration of the graph structure, namely the successor and predecessor functions.

Definition 2.2 (Successors and Predecessors)
Let $\mathcal{M}$ be a finite set of nodes (GENS); A a finite set of attributes; V a finite set of values; $DIFF \subseteq \mathcal{M} \times A \times V \times \mathcal{M}$ a discrimination function.

The successor function Δ is defined as follows:
$\Delta : \mathcal{M} \rightarrow \mathcal{P}(\mathcal{M})$ [1]
$\Delta(M) = \{M' \in \mathcal{M} \mid \exists a \in A, v \in V \,.\, (M, a, v, M') \in DIFF\}$
$\Delta^0(M) = \{M\}$ *and* $\Delta^{(n+1)}(M) = \bigcup\{\Delta(M') \mid M' \in \Delta^n(M)\}$
$\Delta^+(M) = \bigcup_{i=1}^{\infty} \Delta^i(M)$ *(transitive closure)*
$\Delta^*(M) = \bigcup_{i=0}^{\infty} \Delta^i(M)$ *(reflexive and transitive closure)*
In the same way the predecessor function ∇ is defined:
$\nabla : \mathcal{M} \rightarrow \mathcal{P}(\mathcal{M})$
$\nabla(M) = \{M' \in \mathcal{M} \mid \exists a \in A, v \in V \,.\, (M', a, v, M) \in DIFF\}$
$\nabla^0(M) = \{M\}$ *and* $\nabla^{(n+1)}(M) = \bigcup\{\nabla(M') \mid M' \in \nabla^n(M)\}$
$\nabla^*(M) = \bigcup_{i=0}^{\infty} \nabla^i(M)$ *and* $\nabla^+(M) = \bigcup_{i=1}^{\infty} \nabla^i(M)$

Definition 2.3 (Case–Memory)
Let $\mathcal{M}$, A, V, $DIFF$ be as above; $M_0 \in \mathcal{M}$ a special node, the root of the case memory; $NORM \subseteq \mathcal{M} \times A \times V$ a generalization relation.

The 6-tuple $CM = (\mathcal{M}, M_0, A, V, NORM, DIFF)$ is named case memory, *if* $\forall M, M', M'' \in \mathcal{M} \,.\, \forall a \in A \,.\, \forall v \in V$

1. $(M, a, v, M') \in DIFF \wedge (M, a, v, M'') \in DIFF \Rightarrow M' = M''$ *(DIFF is functional)*
2. $(M, a, v, M') \in DIFF \Rightarrow (M', a, v) \in NORM \wedge (M, a, v) \notin NORM$
3. $(M, a, v) \in NORM \wedge (M, a, v') \in NORM \Rightarrow v = v'$
4. $M \notin \Delta^+(M)$ *(the graph is acyclic)*
5. $M \in \Delta^*(M_0)$ *(M_0 is the root of the graph)*

For short spelling (projections):
$NORM_M = \{(a, v) \mid (M, a, v) \in NORM\}$
$DIFF_M = \{(a, v, M') \mid (M, a, v, M') \in DIFF\}$
If $(M, a, v, M') \in DIFF$, then (a, v) is called index for M' in M.

If $(M, a, v) \in NORM$, then most, but not all, successors of M, and M itself, have the value v for attribute a. Further details concerning the construction of the case memory and the insertion of new cases can be found in [Kolodner83a].

If $(M, a, v, M') \in DIFF$, then M' is a successor of M and the link from M to M' is marked with the label (a, v). This means M and M' differ in the value of the attribute a.

If $\Delta(M) = \emptyset$, then the node M is a case, otherwise the node is a GEN.

[1] $\mathcal{P}(\mathcal{M})$ denotes the set of all subsets of $\mathcal{M}$

After this definition of the case memory, it is now possible to say what a generalization really means, especially what information it contains. The closure of a node corresponds to the information we have about it and is defined as the collection of NORMS on a path from the root of CM to this node. NORMS of specialized nodes are preferred over NORMS of generalized nodes. This closure is in general not uniquely determined because there may be several different paths from the root to a node.

We need a function to combine attribute–value pairs. This function builds the union of the set of features with a conflict resolution strategy in the case of common attributes with different values.

Definition 2.4 ($\oplus$)
$\oplus : (A \times V) \times (A \times V) \rightarrow (A \times V)$ is defined as:

$$F \oplus \emptyset = F$$

$$F \oplus (F' \cup \{(a,v)\}) = \begin{cases} F \oplus F' & \text{, if } (a,v') \in F \text{ for some } v' \in V \\ (F \oplus F') \cup \{(a,v)\} & \text{, if } (a,v') \notin F \text{ for all } v' \in V \end{cases}$$

To see what the output of the retrieval process is, we need to know the following notions: path, closure and retrieval. A path is easily defined as a sequence of nodes, while a closure is the collection of information from nodes on such a path.

Definition 2.5 (Path and Closure)
Let $CM = (\mathcal{M}, M_0, A, V, NORM, DIFF)$.
A path P from M to M' in CM is a sequence of nodes $(M_1, ..., M_n)$ with $M_1 = M$ and $M_n = M'$ and $\forall 1 \leq i \leq n-1 \, . \, M_{i+1} \in \Delta(M_i)$

For short spelling:
$\mathcal{P}(M, M') = \{P \mid P \text{ is a path from } M \text{ to } M'\}$

The closure $I(P) \subseteq A \times V$ of a path $P = (M_1, ..., M_n)$ is recursively defined:
$n = 0 \, . \, I(P) = \emptyset$
$n > 0 \, . \, I(P) = NORM(M_n) \oplus I(P')$ *with* $P' = (M_1, ..., M_{n-1})$

The closure of a node M is the collection of the closures of all paths from the root node M_0 to M:
$\mathcal{I}(M) = \{I(P) \mid P \in \mathcal{P}(M_0, M)\}$

Now we will see how information is retrieved from the case memory. For a new case it is necessary to choose the nodes subsuming it (R_{all}) and then to restrict these nodes to the most specialized ones (R).

Definition 2.6 (Retrieval)
Let CM be as above, $\mathcal{F}$ a case base, $F \in \mathcal{F}$ a case.
$R_{all} \subseteq \mathcal{F} \times \mathcal{M}$ is named full retrieval relation *iff*

1. $\forall F \in \mathcal{F} \, . \, (F, M_0) \in R_{all}$
2. $\forall F, F' \in \mathcal{F} \, . \, \forall a \in A \, . \, \forall v \in V \, . \, \forall M, M' \in \mathcal{M} \, .$
   $(F, M) \in R_{all} \wedge (M, a, v, M') \in DIFF \wedge F' = F \cup \{(a,v)\} \Rightarrow (F', M') \in R_{all}$

$R \subseteq R_{all}$ is the retrieval relation iff $\forall (F, M), (F, M') \in R$
$M' \in \Delta^*(M) \Rightarrow M = M'$

The minimal closure *$\mathcal{I}(F)$ of a case F is defined as follows:*
$\mathcal{I}(F) = \{F \oplus I(M) | (F, M) \in R \wedge I(M) \in \mathcal{I}(M)\}$

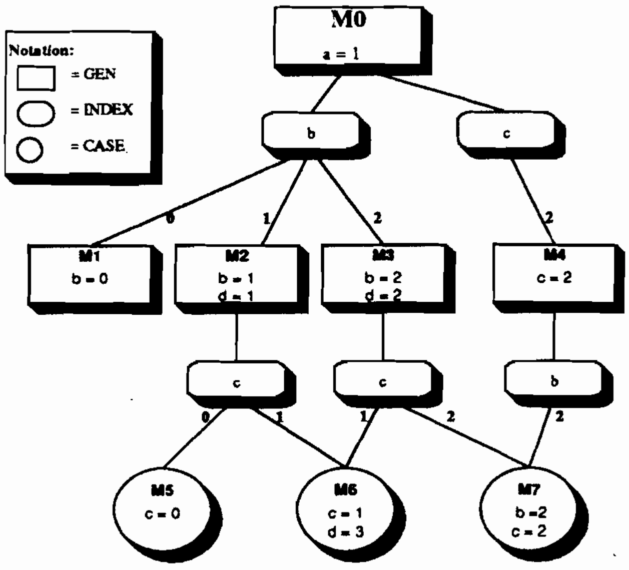


Figure 1: A small case memory

Two indeterminisms are included in the definition of the minimal closure of F: (1) the classification of the case into the case memory and (2) the computing of the closure from different paths.

A small example may help understanding the definitions of cases, case memory and minimal closure: $F = \{(a,1),(b,2),(c,2)\}$ is a typical case. In figure 1 we present a small case memory. For a case $F = \{(b,1)\}$ the retrieval relation R_{all} contains the tuples (F, M_0) and (F, M_2) and R contains (F, M_2) because M_2 is a successor of M_0. Some minimal closures are: $I(M_0) = \{(a,1)\}$ and $\mathcal{I}(M_6) = \{\{(a,1),(b,2),(c,2),(d,2)\},\{(a,1),(b,2),(c,2)\}\}$, i.e. M_6 has two different minimal closures.

2.2 Prioritized Defaults

With the help of defaults we achieve the possibility to represent incomplete and inconsistent knowledge. Default information can be read as "until no more detailed information is available assume that ..." or "if it is consistent to assume that ...".

[Brewka89] orders the defaults into different levels of reliability and computes maximal subsets preferring high and safety levels. These subsets must be consistent, so that defaults at lower levels are eliminated if they lead to inconsistencies.

The ordering of the defaults in levels of reliability can be generalized to partial ordered defaults to model different chains of reliability.

Definition 2.7 (Prioritized default theory)
A prioritized default theory $(\mathcal{T}, \lhd)$ *consists of a set of formula sets* $\mathcal{T} = \{T_i \mid i \in I \subseteq NAT\}$ [2] *with a partial strict well-founded ordering* $\lhd$ *on* $\mathcal{T}$ *and the condition* $\forall i \neq j \,.\, T_i \cap T_j = \emptyset$
$(\mathcal{T}, \lhd)$ *is a* leveled default theory, *if* $\lhd$ *is a total ordering.*

[2]NAT stands for the natural numbers

Intuitively $T_i < T_j$ means that the formulas in T_i are more reliable than these in T_j. There are no preferences among the formulas in one set T_i.

Definition 2.8 (Preferred subtheory)
Let $\Delta = (\mathcal{T}, \lhd)$ be a prioritized default theory with $\mathcal{T} = \{T_i \mid i \in I\}$.
$S \subseteq \mathcal{L}$ is a preferred subtheory *of $\mathcal{T}$ wrt $\lhd$, if the following holds:*

(B1) *It exists a sequence $(t_j \mid j \in J \subseteq NAT)$ with*

1. $\forall i \in I \,.\, \forall t \in T_i \,.\, \exists j \in J \,.\, t = t_j$
2. $\forall j \in J \,.\, \exists i \in I \,.\, t_j \in T_i$
3. $\forall j_1, j_2 \in J \,.\, t_{j_1} \neq t_{j_2}$
4. $\forall t_j \in T_i \,.\, \forall t_{j'} \in T_{i'} \,.\, T_i \lhd T_{i'} \Rightarrow j \leq j'$

(B2) $S = \bigcup_{j \in J} S_j$ *mit* $S_j \subseteq G(t_j)$ [3]

(B3) $\forall k \in J \,.\, \bigcup_{j \leq k} S_j$ *is a maximal, consistent set.*

These preferred subtheories are the basis for reasoning. S is a consistent set of formulas that constitute the current beliefs.

An *extension* E of a prioritized default theory Δ is defined as the set of all formulas that may be derived from a preferred subtheory S. The set of all extensions is denoted by $\mathcal{E}(\Delta)$.

3 Foundation of the case memory using prioritized defaults

With the formal framework from section 2 it is now possible to formulate the main result of the work namely the transformation from the case memory to prioritized defaults.

First we will present an first order language for the description of the case memory.

With the help of prioritized defaults we define a preferred subtheory, such that the extension contains the closure of solution paths encoded in some predicates.

3.1 First order language

A typical scenario for the usage of our new first order language consists of two parts. We have to describe the case memory with formulas and defaults. Furthermore we must encode the input cases into formulas. Then an extension is constructed from which the solutions can be obtained.

The first order language contains the well known logical symbols $\vee, \wedge, \ldots$ and some function and predicate symbols. The nodes, attributes and values are represented as objects in the language. That means for every object the language contains a function symbol with arity zero (a constant).

Beside the function symbols there are predicate symbols which we need to express connections among the objects. A predicate symbol *norm* with arity three is used for the coding of the $NORM$-relation. In the same manner the $DIFF$-relation is coded into a predicate *diff* with

[3] $G(t)$ denotes the set of all ground instances of the formula t

arity four. Furthermore there is a subsumption predicate *sub*, which connects the input cases with the nodes under which they are subsumed. The subsumption will be detailed later. For now the intuitive meaning of subsumption as coding of the solution path is sufficient.

Definition 3.1 (First order language for the case memory)
Let $CM = (\mathcal{M}, M_0, A, V, NORM, DIFF)$ be a case memory and $\mathcal{Q} \subseteq \mathcal{P}(A \times V)$ a set of input cases.

$\mathcal{L}_{CM}$ is a first order language for CM, if

1. *$\mathcal{L}_{CM}$ is a first order language.*
2. *$\forall a \in A$ it exists a constant in $\mathcal{L}_{CM}$ named a.*
3. *$\forall v \in V$ it exists a constant in $\mathcal{L}_{CM}$ named v.*
4. *$\forall M \in \mathcal{M}$ it exists a constant in $\mathcal{L}_{CM}$, named m.*
5. *$\forall Q \in \mathcal{Q}$ it exists a constant in $\mathcal{L}_{CM}$, named q.*
6. *All constant symbols are unique.*
7. *$\mathcal{L}_{CM}$ contains a predicate symbol norm, with arity three, a predicate symbol diff, with arity four, and a predicate symbol sub with arity two.*

A little example should show how the language is used.

Example 3.1
Let $Q = \{(a, 1), (b, 2)\}$ be an incomplete case which is identified with the function symbol q. $a, b, 1, 2$ are further constants of our language $\mathcal{L}_{CM}$.

$norm(q, a, 1) \wedge norm(q, b, 2)$ is an example of a correct formula. The intuitive meaning is that the case Q has the value 1 for the attribute a and 2 as value of b.

Let M be a node of the case memory. More examples of formulas are:

$norm(m, a, 2)$: the attribute a has in node M the value v.

$diff(m, b, 1, m')$: the node M has the successor M' with the index $(b, 1)$.

$sub(m, q)$: the case Q is subsumed by the node M.

3.2 The partial ordered default theory

Now we present how it is possible to use the language $\mathcal{L}_{CM}$ from section 3.1 for the transformation of the case memory and of the input cases into defaults and facts. Additionally priorities for the defaults are given to obtain a partial ordered default theory.

We presuppose a case memory $CM = (\mathcal{M}, M_0, A, V, NORM, DIFF)$ and set of incomplete cases $\mathcal{Q}$. The necessary formulas resp. defaults are introduced one after one. They are collected into a transformation at the end of the section. $\mathcal{T}$ is the name of the set of formulas for describing the case memory.

3.2.1 Formulas and defaults

First we look at the information stored in the nodes. For every node and every attribute value pair in the $NORM$ relation of the node there is an atomic formula containing exactly the triple of node, attribute and value.

(F1) $\forall(M, a, v) \in NORM \,.\, norm(m, a, v) \in \mathcal{T}$

The second step, the transformation of the indices, works exactly in the same manner.

(F2) $\forall(M, a, v, M') \in DIFF \,.\, diff(m, a, v, m') \in \mathcal{T}$

The language $\mathcal{L}_{CM}$ contains a constant symbol for every input case to identify it. The identification is especially necessary for the specification of the input cases. To prevent the introduction of unnecessary predicate symbols, the combination of attribute value pairs with the input cases is also realized with the *norm* predicate.

(F3) $\forall Q \in \mathcal{Q} \,.\, \forall(a, v) \in Q \,.\, norm(q, a, v) \in \mathcal{T}$

The identification of the input cases is also necessary for the definition of the subsumption relation. We will describe a solution path with the set of nodes subsuming an incomplete case. Because every path starts at the root, every case must be subsumed from it.

(F4) $\forall Q \in \mathcal{Q} \,.\, sub(m_0, q) \in \mathcal{T}$

To build the subsumption relation it is necessary to traverse indices and to generate new nodes that subsume the input cases. Primary the default rule $spec(x_m, x_a, x_v, x_q)$ is given, which is explained below.

(D5) $\forall M \in \mathcal{M} \,.$

$spec_M(x_m, x_a, x_v, x_q) : diff(m, x_a, x_v, x_m) \wedge sub(m, x_q) \wedge norm(x_q, x_a, x_v) \Rightarrow sub(x_m, x_q) \in \mathcal{T}$

The rule tells in principle: if we know there is a link between a node M and some successor node x_m with the index (x_a, x_v) $(diff(m, x_a, x_v, x_m))$ and a case x_q, which has the value x_v for the attribute x_a $(norm(x_q, x_a, x_v))$, is subsumed by M $(sub(m, x_q))$, then the case is also subsumed by x_m $(sub(x_m, x_q))$.

Now we describe what it means that a case is subsumed by a node. The intuitive meaning is that the information stored in the NORM relation of a node is transferred to the case, that is subsumed by the node.

(D6) $\forall M \in \mathcal{M} \,.\, sol_M(x_q, x_a, x_v) : sub(m, x_q) \wedge norm(m, x_a, x_v) \Rightarrow norm(x_q, x_a, x_v) \in \mathcal{T}$

Two formulas are still missing. They guarantee the existence of exactly one solution path and at most one value for every attribute in the solution. These two formulas are especially interesting, because they may lead to inconsistencies among the defaults, which we have studied until now.

The following formula means that it is impossible that there are two values for one attribute in a case.

(F7) $norm(x_q, x_a, x_v) \wedge norm(x_q, x_a, x_w) \Rightarrow x_v = x_w \in \mathcal{T}$[4]

Another demand is that two nodes cannot subsume an input case if they are not on a common path:

(F8) $\forall M, M' \in \mathcal{M}$ with $M' \notin \Delta^*(M) \cup \nabla^*(M) \,.\, \neg sub(m, x_q) \vee \neg sub(m', x_q) \in \mathcal{T}$

The formulation of the conditions seems a little bit tricky, but this condition is necessary to avoid that nodes from different paths subsume the input case. The condition $M' \notin \Delta^*(M) \cup$

[4] $=$ denotes the syntactical equality

$\nabla^*(M)$ means that two nodes which are not in a predecessor or successor relation, cannot subsume a case together.

The following problem is not covered with the rule (F8):

Let CM be a case memory with $M_1, M_2 \in \Delta(M_0)$ and $M_2 \in \Delta(M_1)$ according to figure 2.

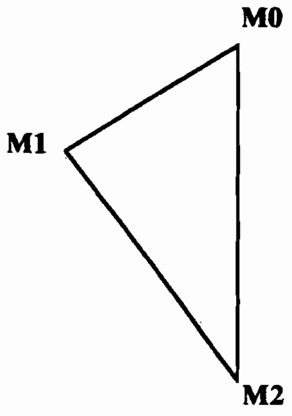


Figure 2: Example for the necessity of default (D9)

Possible solution paths might be (M_0, M_1, M_2) and (M_0, M_2). Because the extension of the default theory computes the maximal set of formulas, we obtain only one of the two possible solutions with the defaults and facts mentioned so far.

To get the second solution we need a default that makes M_1 and M_2 in our example in a common solution path inconsistent.

(D9) $\forall M \in \mathcal{M} \,.\, \forall M', M'' \in \Delta(M) \,.\, M' \neq M''$
$exc_{M,M',M''}(x_q) : \neg sub(m', x_q) \vee \neg sub(m'', x_q) \in \mathcal{T}$

In connection with default (D5) there may be three possible solutions: the two defaults

$spec_{M_0}(m_1, a, v, q), spec_{M_0}(m_2, b, w, q)$ and $exc_{M_0,M_1,M_2}(q)$

are inconsistent and one of them should not be in the extension. So we obtain the following solution paths: $(M_0, M_2), (M_0, M_1), (M_0, M_1, M_2)$. With the correct priorities, we can exclude the path (M_0, M_1) from the solutions.

We call the formulas (F7), (F8) and the default (D9) *exclusion conditions.*

3.2.2 Priorities

The next step towards a partial ordered default theory according to section 2.2 requires the determination of priorities.

We construct sets $T_i (0 \leq i \leq 2n)$ of formulas and define a partial ordering on these T_i.

The highest priorities are assigned to the formulas (F1) – (F4) and (F7) – (F8) which we call facts. These formulas constitute the set T_0 of sure formulas.

We can show that T_0 is consistent. That is the reason why non of the formulas contained in T_0 must be retracted in the construction of a preferred subtheory.

The priorities among the defaults (D5),(D6) and (D9) will be extracted from the algorithm for the retrieval. The retrieval process computes in the first step a solution path. During this computation the case memory is explored beginning at the root node. Corresponding to this behavior the defaults (D5) and (D9) have a higher priority than the defaults (D6).

Not all defaults of the form $spec_M(x_m, x_a, x_v, x_q)$ wrt. $exc_{M,M',M''}(x_q)$ have assigned the same priority, but the priority decreases from the root node to the leafs.

Let n be the number of nodes in the case memory and $\mathcal{M} = \{M_1, ..., M_n\}$ a numbering of the nodes. The sets $T_1, ..., T_n$ are defined as $T_i = \{spec_{M_i}(x_m, x_a, x_v, x_q), exc_{M_i,M',M''}(x_q)\}(1 \leq i \leq n)$.

The partial order among the sets T_i is defined according to the successor relation.

$\forall 1 \leq i,j \leq n \; . \; T_i \lhd T_j$ iff $M_j \in \Delta^+(M_i)$

This means: T_i has a higher priority as T_j if the node M_i is above M_j in the case memory. The partial order is well defined because the successor relation Δ^+ is transitive. It is furthermore anti-symmetrical, because the case memory has no cycles.

T_0 has a higher priority than all $T_i (i \geq 1)$:

$\forall 1 \leq i \leq n \; . \; T_0 \lhd T_i$

The last part of the retrieval algorithm is concerned with the computation of the closure of the solution path. Roughly speaking this is also the idea behind the default (D6). That is the reason why the priorities for the defaults $sol_M(x_q, x_a, x_v)$ are at the lowest level.

Each of the sets $T_{n+1}, ..., T_{n+n}$ contains one defaults $T_{n+i} = \{sol_{M_i}(x_q, x_a, x_v)\}$ with the same numbering of the nodes as above.

The ordering on the sets $T_{n+1}, ..., T_{n+n}$ is reverse to the partial order on $T_1, ..., T_n$, because the information in leaf nodes has precedence over information stored in inner nodes

$\forall 1 \leq i,j \leq n \; . \; T_{n+i} \lhd T_{n+j}$ iff $M_i \in \Delta^+(M_j)$

Because we want to assign the lowest priorities to the sets defined last, we complete our partial order with the following condition:

$\forall 1 \leq i,j \leq n \; . \; T_i \lhd T_{n+j}$

A little example seen in figure 3, gives some illustration of the priorities.

3.2.3 Connection between solutions and extensions

Before we present the whole transformation of a case memory into a partial ordered default theory completely some remarks about the extraction of the solution from extensions should be made. The formal connection between solutions computed by the retrieval process and extensions will be given later on in the correctness and completeness theorems.

To obtain a solution for an input case Q, we look at the set of formulas that can be deduced from the facts and defaults of a preferred subtheory. This set of formulas is called the extension. The solution can be found in a subset of the extension containing atomic formulas of the form *norm*$(q, -, -)$, which have the constant q for the identification of the case Q as first argument. The conjunction of these formulas describes exactly a solution.

Many extensions can be computed, because many different subtheories are possible. We show below that the different extensions exactly match the different solutions of the retrieval process.

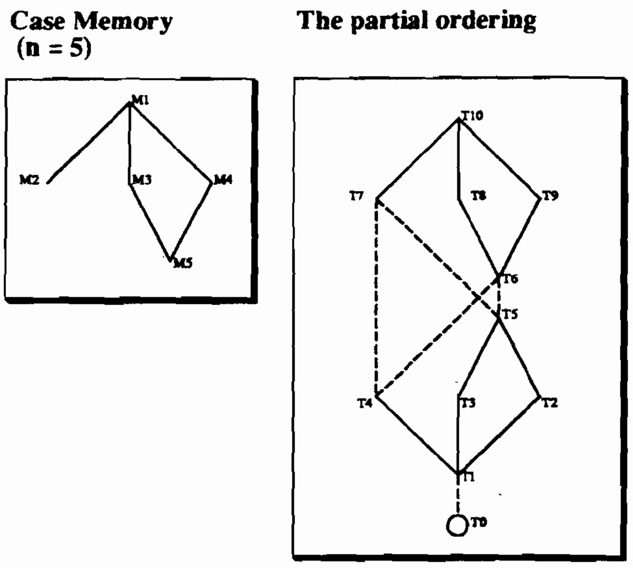


Figure 3: Priorities

3.2.4 The complete transformation

The transformation of the case memory into a default theory $\Delta = (\mathcal{T}, \lhd)$ consists of the definition of a set of formula sets $\mathcal{T} = \{T_0, ..., T_{n+n}\}$ and a strict partial ordering $\lhd$ on $\mathcal{T}$.

Transformation 3.1 (Partial ordered default theory for a case memory)
Let $CM = (\mathcal{M}, M_0, A, V, NORM, DIFF)$ be a case memory, $\mathcal{Q} \subseteq \mathcal{P}(A \times V)$ a set of incomplete cases, i.e. $\forall Q \in \mathcal{Q} . Q$ is a partial function: $A \rightarrow V$, $\mathcal{L}_{CM}$ a first order language and n the numbers of nodes in $\mathcal{M}$.

$\Delta = (\mathcal{T}, \lhd)$ is a partial ordered default theory of the case memory CM wrt. $\mathcal{Q}$, if

1. $\mathcal{T} = \{T_0, ..., T_{n+n}\}$
2. *T_0 contains the following formulas*
   (a) (F1) $\forall (M, a, v) \in NORM . norm(m, a, v) \in T_0$
   (b) (F2) $\forall (M, a, v, M') \in DIFF . diff(m, a, v, m') \in T_0$
   (c) (F3) $\forall Q \in \mathcal{Q} . \forall (a, v) \in Q . norm(q, a, v) \in T_0$
   (d) (F4) $\forall Q \in \mathcal{Q} . sub(m_0, q) \in T_0$
   (e) (F7) $norm(x_q, x_a, x_v) \land norm(x_q, x_a, x_w) \Rightarrow x_v = x_w \in T_0$
   (f) (F8) $\forall M, M' \in \mathcal{M}$ *with* $M' \notin \Delta^*(M) \cup \nabla^*(M) . \neg sub(m, x_q) \lor \neg sub(m', x_q) \in T_0$
3. *The sets of defaults T_i $(1 \leq i \leq n)$ are defined as*
   (D5),(D9) $T_i = \{$
   $spec_{M_i}(x_m, x_a, x_v, x_q) : diff(m_i, x_a, x_v, x_m) \land sub(m_i, x_q) \land norm(x_q, x_a, x_v)$
   $\Rightarrow sub(x_m, x_q)$,
   $exc_{M_i, M', M''}(x_q) : \neg sub(m', x_q) \lor \neg sub(m'', x_q)\}$
4. *The sets of defaults T_i $(n + 1 \leq i \leq n + n)$ are defined as*

(D6) $T_i = \{sol_{M_i}(x_q, x_a, x_v) : sub(m_i, x_q) \wedge norm(m_i, x_a, x_v) \Rightarrow norm(x_q, x_a, x_v)\}$

5. *The strict partial order* $\lhd$ *of the* T_i *is defined as*

   (a) $\forall 1 \le i \le n \,.\, T_0 \lhd T_i$

   (b) $\forall 1 \le i, j \le n \,.\, T_i \lhd T_{n+j}$

   (c) $\forall 1 \le i, j \le n \,.\, T_i \lhd T_j$ *iff* $M_j \in \Delta^+(M_i)$

   (d) $\forall 1 \le i, j \le n \,.\, T_{n+i} \lhd T_{n+j}$ *iff* $M_i \in \Delta^+(M_j)$

3.3 Propositions for the transformation

This section contains some theorems about the transformation 3.1. Fundamental results are the theorems about correctness and completeness of the partial ordered default theory wrt. the case memory.

The first proposition shows the consistency of the ground terms constructed from the facts.

Lemma 3.1 (Consistency of $G(T_0)$)
Let Δ be as in transformation 3.1.

$G(T_0)$ *is consistent*

This proposition has a nice conclusion: every preferred subtheory of the partial ordered subtheory Δ contains $G(T_0)$. The reason therefor is the consistency and the priority of the defaults in T_0.

Conclusion 3.1
Let CM, Q, Δ be as in transformation 3.1.

For all extensions E of Δ . $G(T_0) \in E$.

For a nice formulation of the following theorems we need two more notations: *compatibility of a solution path with an extension* and *compatibility of a solution with an extension.*

Compatibility of a solution path with an extension means that the solution path can be found in the subsumption relation (*sub*) of the extension. Compatibility of a solution with an extension has the analogous meaning that the solution can be found in the *norm*-relation.

Definition 3.2 (Compatibility)
Let CM, Q, Δ be as in transformation 3.1.

We call a solution path $LP(Q)$ and an extension E compatible *if:*

$\forall M \in \mathcal{M} \,.\, M \in LP(Q)$ *iff* $sub(m, q) \in E$

We call a solution $L(Q)$ and an extension E compatible, *if:*

$\forall a \in A \,.\, \forall v \in V \,.\, (a, v) \in L(Q)$ *iff* $norm(q, a, v) \in E$

These compatibility definitions show how the retrieval in the case memory and the computation of extensions can be brought together.

The following theorem of correctness states that every extension is compatible with some solution of the retrieval in the case memory.

Theorem 3.1 (Correctness)
Let $CM, \mathcal{Q}, \Delta$ be as in transformation 3.1.

$\forall Q \in \mathcal{Q} \,.\, \forall E \in \mathcal{E}(\Delta) \,.\, \exists L(Q) \in \mathcal{L}(Q)$

$\quad \forall a \in A \,.\, \forall v \in V \,.\, norm(q,a,v) \in E \text{ iff } (a,v) \in L(Q)$.

Analogously we see the completeness of our transformation: every solution of the retrieval process has an compatible extension.

Theorem 3.2 (Completeness)
Let $CM, \mathcal{Q}, \Delta$ be as in Transformation 3.1.

$\forall Q \in \mathcal{Q} \,.\, \forall L(Q) \in \mathcal{L}(Q) \,.\, \exists E \in \mathcal{E}(\Delta)$

$\quad \forall a \in A \,.\, \forall v \in V \,.\, (a,v) \in L(Q) \text{ iff } norm(q,a,v) \in E$

To show this two strong results we use the following two weaker lemmata about solution paths.

Lemma 3.2 (Completeness of solution paths)
Let $CM, \mathcal{Q}, \Delta$ be as in transformation 3.1.

$\forall Q \in \mathcal{Q} \,.\, \forall LP(Q) \in \mathcal{LP}(Q) \,.\, \exists E \in \mathcal{E}(\Delta)$ *with*

1. $\forall M \in \mathcal{M} \,.\, M \in LP(Q) \leadsto sub(m,q) \in E$
2. $\forall M \in \mathcal{M} \,.\, M \notin LP(Q) \leadsto sub(m,q) \notin E$

Lemma 3.3 (Correctness of solution path)
Let Δ be as in transformation 3.1.

$\forall Q \in \mathcal{Q} \,.\, \forall E \in \mathcal{E}(\Delta) \,.\, \exists LP(Q) \in \mathcal{LP}(Q)$ *with*

1. $\forall sub(m,q) \in \mathcal{L}_{CM} \,.\, sub(m,q) \in E \leadsto M \in LP(Q)$
2. $\forall sub(m,q) \in \mathcal{L}_{CM} \,.\, sub(m,q) \notin E \leadsto M \notin LP(Q)$

4 Discussion

The current work in the field of CBR research lacks the exploration of the logical basis. Non-monontonic reasoning provides especially for CBR an adequate foundation, because additional information about cases leads to revision of inferred knowledge.

This work supplies a first formal definition and foundation of the case memory using prioritized defaults. Using these defaults is obvious because there are already preference relations between the nodes encoded in the representation of the case memory as directed acyclic graph.

The first attempt to model the case memory with defaults is the work [KotonChase89]. But in several aspects this paper is inadequate and incomplete. It lacks the exact definition of the structure of the case memory because the defaults mentioned cannot represent the case memory in general. Another problem is the absence of theorems about the usefulness of the defaults. One major point of criticism is that the complexity of the defaults is very high, if the case memory has a large number of indices. Our work provides smaller defaults, because we need to encode only local information, while the global control is encoded into priorities.

Acknowledgements

My work originate in the frame of the MOLTKE-project [Richter91]. MOLTKE is an expert system shell, in which components for case-based reasoning are included (GENRULE and PATDEX). One application domain for the shell is the diagnosis of faults in a CNC-center. For this application there exists a voluminous case base. This work is part of the MOLTKE-project. MOLTKE [Richter91] is a workbench for technical diagnosis which was developed at the university Kaiserslautern over the past years and contains, among others, components for case-based reasoning (PATDEX and GENRULE).

I would like to thank professor Michael M. Richter and Ulrich Junker for useful discussion, motivation and many valuable hints. Also Stefan Wess and the other members of our group have made many constructive criticism in every phase of this work.

References

[Brewka89] Gerhard Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the Eleventh International Joint Conference on Artifical Intelligence*, pages 1043 - 1048. IJCAI, 1989.

[KotonChase89] Phyllis Koton and Melissa P. Chase. Knowledge representation in a case-based reasoning system: Defaults and exceptions. In Ronald J. Brachman and Hector J. Levesque, editors, *Proceedings of the First International Conference on Priciples of Knowledge Representation and Reasoning*, pages 203 - 211. Morgan Kaufmann Publishers, Inc, 1989.

[Kolodner83a] Janet L. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243-280, 1983.

[Kolodner83b] Janet L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7:281-328, 1983.

[Kolodner84] Janet L. Kolodner. *Retrieval and organizational strategies in conceptual memory: A computer model.* Erlbaum, 1984.

[Reiter80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81 - 132, 1980.

[Richter91] Michael M. Richter, editor. *Das MOLTKE - Buch.* Springer, To appear 1991.

[RisslandKolodnerWaltz89] Edwina Rissland, Janet Kolodner, and David Waltz. Proceedings from the case-based reasoning workshop, pensacola beach, florida. In Edwina Rissland, Janet Kolodner, and David Waltz, editors, *Proceedings from the Case-Based Reasoning Workshop, Pensacola Beach, Florida*, pages 1 - 13. DARPA, Morgan Kaufmann, 1989.

[Schank82] Roger C. Schank. *Dynamic Memory: A theory of reminding and learning in computers and people.* Cambridge University Press, 1982.

[Slade91] Stephen Slade. Case-based reasoning: A research paradigm. *AI Magazine*, 1:42 - 55, 1991.