An Application of Terminological Logics to Case-based Reasoning*

Jana Koehler
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3,
D-66123 Saarbrücken, Germany
e-mail: koehler@dfki.uni-sb.de

Abstract

A key problem in case-based reasoning is the representation, organization and maintenance of case libraries. While current approaches rely on heuristic and psychologically inspired formalisms, terminological logics have emerged as a powerful representation formalism with clearly defined formal semantics.

This paper demonstrates how the indexing of case libraries can be grounded on terminological logics by using them as a kind of query language to the case library. Indices of cases are represented as concepts in a terminological logic. They are automatically constructed from the symbolic representation of cases with the help of a well-defined abstraction process. The retrieval of cases from the library is grounded on concept classification.

The theoretical approach provides the formal foundation for the fully implemented case-based planning system MRL. The use of terminological logics allows formal proof of properties like the correctness, completeness and efficiency of the retrieval algorithm, which has rarely been done for existing case-based reasoning systems.

1 INTRODUCTION

Reasoning from second principles has emerged as a new research paradigm in problem solving. Instead of searching a solution by reasoning from scratch, this method bases the entire problem-solving process on the reuse and modification of previous solutions.

Current approaches are within the field of *case-based reasoning* which is defined as a general paradigm for reasoning from experience [Slade, 1989]. Approaches to case-based reasoning rely mainly on psychological theories of human cognition and have led to a wide variety of proposals for the representation, indexing and organization of case libraries, cf. [CBR-91, 1991; CBR-93, 1993].

Case-based systems reason by *approximation* and *similarity*. In order to solve a new case by reusing an existing one from a *case library*, several reasoning tasks have to be addressed: First, an *index* is derived from the new case by extracting those of its features that are *abstract enough* to make a case useful in a variety of situations as well as *concrete enough* to be easily recognizable in future situations. In most approaches, the *index vocabulary* is a subset of the vocabulary used for the symbolic representation of cases, cf. [Kolodner, 1993].

The index of the new case is used as a search key on which the *retrieval* of applicable old cases is based. The aim of retrieval is to determine "good cases" *efficiently* in the library—those that make relevant predictions about the current case. Besides the need for an efficient search strategy, the retrieval problem implies the *matching problem*, which is a serious bottleneck for case-based reasoners, cf.[Kolodner, 1993; Slade, 1989; Riesbeck and Schank, 1989]. As we cannot expect that features of different cases coincide completely, so-called *partial matches* have to be computed and cases with *best-matching indices* have to be retrieved. Finally, the set of retrieved cases is ordered according to *ranking heuristics* and the "best" case is determined.

Research in case-based reasoning proposes various solutions to the problems of retrieval, indexing and matching. A common characteristic of these solutions is that they are described in an *informal* way. This makes it difficult to compare the various approaches, to prove their formal properties and to extend them to other applications.

Nevertheless, practice imposes the following require-

*This paper has been published in the Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning, pages 351–362, Ed. by J. Doyle, and E. Sandewall, and P. Torasso, Morgan Kaufmann, San Francisco 1994.

ments on case-based reasoners:

- The behavior of a system should be predictable. It should be possible to verify whether the system implements the intended behavior correctly.
- The derivation of *indexes* should be done automatic instead of "hand-coded", which is still usual in many approaches.
- The retrieval algorithm should find a solution to a new case, if this solution exists in the case library.
- If no direct solution can be determined, the retrieval algorithm should determine the case that best meets the search criterion.

Consequently, this implies the need for case-based reasoning systems with *formal semantics*. The retrieval algorithm should have the following formal properties:

- **Correctness**: The retrieved case is guaranteed to meet the search criterion.
- **Completeness**: Retrieval of existing solutions to new cases from the case library is ensured.
- **Complexity**: The retrieval algorithm is proved to be efficient, i.e. it runs in polynomial time.

A case-based reasoning system with these properties can be expected to meet the challenge of *scaling-up*: The system's behavior remains predictable, sound and efficient even when it is applied to large-scale real-world problems. Surprisingly, it turns out that problems like the correctness and completeness of the retrieval algorithm have not been widely discussed in the literature on case-based reasoning.

The work described in this paper is motivated by research in *case-based planning*. The reuse and modification of plans is a valuable tool for improving the efficiency of planning, because it avoids the repetition of planning effort. Therefore, plans that have been obtained as solutions for planning problems are stored in *plan libraries* for further use. The retrieval of a good plan from a plan library is identified as being a serious bottleneck for plan reuse systems in [Nebel and Koehler, 1993a; Nebel and Koehler, 1993b]. Consequently, efficient and theoretically well-founded retrieval and update procedures for plan libraries have to be developed.

The approach presented in this paper suggests the integration of terminological logics into a hybrid representation formalism for case-based reasoning. Retrieval from and updating of case libraries are grounded on a clearly defined formalism with proper semantics. Their behavior becomes predictable and formal properties like the completeness and soundness of the retrieval algorithm can be proved.

2 THE SOLUTION

While case-based reasoning aims at developing a scientific model of human memory, research in *knowledge representation and reasoning* has led to concept languages of the KL-ONE family [Brachman, 1978], also called *terminological logics*. Terminological logics support a *structured* representation of *abstract* knowledge. In contrast to earlier representation formalisms, terminological logics possess formal semantics. The Tarski style declarative semantics leads them to be considered as sublanguages of predicate logic [Brachmann and Levesque, 1984]. With that, the meaning of expressions within the formalism is clearly defined and it is possible to verify whether or not the knowledge-representation system correctly implements the intended behavior. Furthermore, terminological logics provide special-purpose inference algorithms like *subsumption* and *classification*. These properties of terminological logics clearly suggest their use in case-based reasoning.

2.1 FORMALIZING CASE-BASED REASONING

A *case* represented in a case library consists of three major parts, cf. [Kolodner, 1993]:

- **initial situation**: A state description, pre, specifying the preconditions, on which the solution represented in the case relies.
- **resulting situation**: The goal state, $goal$, that is achieved when the solution is carried out.
- **solution**: A solution S that solves the problem specification of the case $C = \langle pre, goal \rangle$.

Case-based reasoning starts with a new case in the form of a problem specification

$$C_{new} = \langle pre_{new}, goal_{new} \rangle$$

for which a solution has to be found in the case library.

- **Given:** a new case C_{new}
- **Wanted:** a solution S_{old} from the case library

To find this solution, a *search key* is derived from the problem specification, which has to reflect the main properties of the problem. Usually, the search is done in the *state space* rather than the solution space. This means, instead of searching the case library directly for solutions, it is searched for *similar* problem specifications. This is justified by the following observation: the solution S_{old} is the result of a previous problem-solving process, i.e. it solves an old case C_{old} in the sense that

$$S_{old} \models C_{old}$$

This means, a solution which is applied in an initial situation satisfying *pre* achieves a resulting situation satisfying *goal*. This suggests a search of the case library for previous problem specifications, i.e. old cases, which entail the problem specification of the new case in the sense that each solution for C_{old} is a solution for C_{new}:

$$C_{old} \models C_{new}$$

If this relationship between C_{old} and C_{new} holds, then the new case has been shown to be an instance of a case from the library. This implies that solving C_{old} is *sufficient* for solving C_{new}. Consequently, the solution S_{old} stored in C_{old} will solve C_{new}.

2.2 REASONING BY APPROXIMATION

Searching a case library according to the $\models$ relationship is obviously too restrictive. Such a search algorithm would only retrieve solutions from the case library. But obviously, a "good" case is one that can be easily adapted to obtain the desired solution. Furthermore, the retrieval process is based on an *index* that is obtained from C_{new} instead of directly taking the specification of C_{new}. Therefore, an *index* of a case is computed with the help of an *encoding scheme* ω mapping the case

$$C_{new} = \langle pre_{new}, goal_{new} \rangle$$

to its index

$$\omega(C_{new}) = \langle \omega(pre_{new}), \omega(goal_{new}) \rangle$$

The encoding scheme formalizes an *abstraction process*: A detailed specification of a particular case is mapped to an abstract index reflecting the main features of that case. The degree of abstraction is determined by the particular encoding scheme, which is used in the case-based reasoning system. This means that different encoding schemes can define different degrees of abstraction in a case-based reasoning system.

The encoding scheme ω has to possess the following formal property:

$$\text{If } C_{old} \rightarrow C_{new} \text{ then } \omega(C_{old}) \rightarrow \omega(C_{new})$$

This theorem gives a *monotonicity property* of ω. An existing subset relationship between the models of the cases C_{old} and C_{new} is preserved as a subset relationship between the models of the indices $\omega(C_{old})$ and $\omega(C_{new})$.

$$\text{If } M_{C_{old}} \subseteq M_{C_{new}} \text{ then } M_{\omega(C_{old})} \subseteq M_{\omega(C_{new})}$$

This *monotonicity property* of the encoding scheme ensures that an existing solution can be found by searching the case library along the $\models$ dimension between indices. Note that the inverse of the monotonicity property does not hold in general. A case retrieved from the library, the index of which entails the new index, will not, with certainty, provide a solution to the new case. This reflects *reasoning by approximation*. The retrieval algorithm approximates the $\models$ relationship between the cases when it compares the indices of the cases. Thereby, it extends the solution set computed by the retrieval algorithm.

The definition of a particular encoding scheme depends on three factors:

- the representation formalism for the cases,
- the representation formalism for the indices,
- the application domain.

In Section 3, we illustrate the definition of an encoding scheme for a *case-based planning system*. The representation formalism for the cases is a *temporal planning logic*. The representation formalism for the indices is a *terminological logic*. The application domain comprises planning tasks arising in a subset of the UNIX operating system.

2.3 REASONING BY SIMILARITY

The second aspect of case-based reasoning is *reasoning by similarity*. Case-based systems compute the similarity of cases by comparing the placement of the cases in the abstraction hierarchy or by computing their distance on a qualitative or quantitative scale, cf. [Kolodner, 1993]. A formalization of the notion of *similarity* is beyond the scope of this paper. Nevertheless, the encoding scheme allows to define when a case is more *specific* than another one:

Definition 1 *A case C_1 is defined as being more specific than a case C_2, if $\omega(C_1) \models \omega(C_2)$ holds for their indices.*

Remember, that a case contains three major parts: its initial situation, its resulting situation and the solution. The entailment relation between cases can therefore be reduced to relations between initial and resulting situations. A case is an instance of a stored case if

- the new initial situation entails the old initial situation $pre_{new} \models pre_{old}$, i.e. the solution S_{old} is applicable to the new initial situation,
- the old resulting situation entails the new resulting situation $goal_{old} \models goal_{new}$, i.e. S_{old} solves at least the new problem.

Furthermore, each index $\omega(C) = \langle \omega(pre), \omega(goal) \rangle$ comprises two components, namely the encoding of the initial situation, *pre*, and the encoding of the resulting situation *goal*. Obviously, testing $\omega(C_{old}) \models \omega(C_{new})$ can be reduced to computing relations between the encodings of both situations:

$$\omega(pre_{new}) \models \omega(pre_{old}) \text{ and } \omega(goal_{old}) \models \omega(goal_{new})$$

Strong and weak retrieval algorithms can thus be defined. A strong retrieval algorithm determines reusable cases by testing

$$\omega(pre_{new}) \models \omega(pre_{old}) \text{ and } \omega(goal_{old}) \models \omega(goal_{new})$$

This guarantees that existing solutions can be found in the case library. Furthermore, *more specific* cases are retrieved according to definition 1.

If strong retrieval fails to find a more specific case, the search criterion is replaced by a weaker one: A weak retrieval algorithm can test

$$\omega(pre_{new}) \models \omega(pre_{old}) \text{ or } \omega(goal_{old}) \models \omega(goal_{new})$$

Thus, we can ground the retrieval of cases on different well-defined relations between indices that possess formal semantics. This overcomes the problem of defining *partial matches* between cases, the semantics of which remains often unclear.

2.4 HYBRID REPRESENTATION

In this paper, we propose a hybrid representation formalism for case libraries: The major parts of a case, the *case entry*, are represented in a formalism that adequately represents problems and solutions in the underlying application domain, e.g. the planning formalism used by a case-based planner.

The *index* of the case is represented as a *concept* in a terminological logic. The relation $\models$ between indices is determined by computing the *subsumption* relation (denoted with $\sqsubseteq$) between concepts. With that, the retrieval of a reusable case from the case library can be grounded on *concept classification*.

The encoding scheme defines the *degree of abstraction* that is reflected in the indices: a given case is mapped to an index reflecting the main properties of the case. Note that the encoding scheme may map several specific cases to the same index. This means, the index represents a description of an abstract class of specific cases occurring in a particular application domain. The case related to this index represents one possible specific instance of that class.

Terminological logics have the following advantages:

They provide indices with clearly defined semantics. The monotonicity property of the encoding scheme ω can be proved. The encoding scheme implements a representational shift from the vocabulary for the symbolic representation of cases to the indexing vocabulary represented in a terminological logic. This leads to a well-defined abstraction process. Furthermore, the indexing vocabulary can be automatically built by the case-based reasoning system: If the vocabulary for the symbolic representation of cases is a logical formalism, a case will be represented as a formula in this logic. The index of the case is obtained by a transformation of the formula with the help of the encoding scheme. The result is a first-order logic formula that can be interpreted as a concept definition in the terminological logic, cf. Section 3.

The mathematical properties of various terminological logics are well understood. In particular, terminological languages with decidable subsumption relations have been identified. Remember, that retrieval from case libraries must be efficient, i.e. the complexity of the retrieval algorithm must be investigated. The use of a terminological logic with a polynomial subsumption algorithm ensures that the retrieval algorithm runs in polynomial time as well.

Most of the indexing schemes used in case-based reasoning, for example *discrimination networks* [Feigenbaum, 1963], restrict the case library to have a *tree* structure. In using terminological logics, case libraries are indexed on a more general *lattice structure* provided by the subsumption hierarchy.

3 AN EXAMPLE

The MRL system [Koehler, 1994a] is the case-based planning component of the system PHI [Bauer *et al.*, 1993], a logic-based tool for intelligent help systems. PHI integrates plan generation as well as plan recognition. Plan generation can be done from first principles by planning from scratch and from second principles by reusing previously generated plans with MRL [Biundo *et al.*, 1992]. The example application domain of PHI is the UNIX *mail domain* where objects like *messages* and *mailboxes* are manipulated by actions like *read*, *delete*, and *save*.

3.1 THE PLANNING LOGIC

The logical basis of PHI and MRL is the interval-based modal temporal logic LLP [Biundo and Dengler, 1994]. LLP provides the modal operators $\circ$ (next), $\Diamond$ (sometimes), $\Box$ (always) and ; (chop), the binary modal operator, which expresses the sequential composition of formulae. As in programming logics, control structures like iterations and conditionals and *local variables* are available, with values that may vary from state to state.

Plans are represented by a certain class of LLP formulae. They may contain, e.g. basic actions which are expressed by the *execute* predicate ex, the *chop* operator, which is used to express the sequential composition of plans, and control structures.

The atomic actions available to the planner are the elementary commands of the UNIX mail system. They are axiomatized like assignment statements in programming logics. State changes which are caused by exe-

cuting an action are reflected in a change of the values of local variables which represent the mailboxes in the mail system. For example, the axiomatization of the *delete*-command which deletes a message x in a mailbox *mbox* reads

$$\begin{array}{l}\forall x\, open\_flag(mbox) = T \wedge \\ \quad delete\_flag(msg(x, mbox)) = F \wedge \\ \qquad ex(delete(x, mbox)) \\ \rightarrow \circ\, delete\_flag(msg(x, mbox)) = T\end{array}$$

The state of a mailbox is represented with the help of *flags*. The precondition of the *delete*-command is that the mailbox *mbox* is open, i.e. its *open_flag* yields the value *true* (T) and that the message x has not yet been deleted, i.e. its *delete_flag* yields the value *false* (F). As an effect, the action sets the *delete_flag* of message x in mailbox *mbox* to the value *true* in the next state.

Planning problems are represented with the help of formal plan specifications in the logic LLP. They contain the specification of an initial state, the *preconditions* of the plan, and the specification of the *goals* that have to be achieved by executing the plan.

As an example, assume that a plan **P1** for the planning problem "read and delete a message m in the mailbox *mybox*" has to be found. As preconditions, we assume that the mailbox *mybox* has already been opened and that the message m has not yet been deleted. The formal specification of the preconditions pre_{P1} and the goals $goal_{P1}$ in the logic LLP reads as follows:

$$\begin{array}{l}pre_{P1}\colon\ open\_flag(mybox) = T \wedge \\ \qquad delete\_flag(msg(m, mybox)) = F\end{array}$$

$$\begin{array}{l}goal_{P1}\colon\ \Diamond[read\_flag(msg(m, mybox)) = T \wedge \\ \qquad\quad \Diamond[delete\_flag(msg(m, mybox)) = T\,]\,]\end{array}$$

It should be noted that in using the logic LLP in a planning system it becomes possible to specify temporary goals with the help of nested *sometimes* operators, i.e. goals that have to be achieved at some point and not necessarily in the end, something which could not be done in the usual STRIPS or TWEAK type planning systems, cf. [Kautz and Selman, 1992]. In the example, the goal specification requires the message to be read first and then deleted.

The plan **P1**, which solves this planning problem, is a simple sequence containing the actions *type* and *delete*:

P1: $ex(type(m, mybox)); ex(delete(m, mybox))$

To obtain this plan by case-based planning in MRL, appropriate candidate plans have to be retrieved from the plan library. In the example, we assume that the plan library contains the candidate plans **P2** and **P3**:

P2: **if** $open\_flag(mbox) = T$
then $ex(empty\_action)$
else $ex(mail(mbox))$;
$ex(type(x, mbox)); ex(delete(x, mbox))$

P3: $n := 1$;
while $n < length(mbox)$ **do**
if $sender(msg(n, mbox)) = joe$
then $ex(type(n, mbox))$;
$ex(delete(n, mbox))$
else $ex(empty\_action)$;
$n := n + 1$
od ;

The plan **P2** is an example of a *conditional* plan. It contains a case analysis on the state of the mailbox *mbox*: If the mailbox is open, the message x can be read and deleted. If the mailbox is closed, we first have to open it before the plan can be executed. The case analysis results from incomplete information about the preconditions for plan **P2**:

$$pre_{P2}\colon\ delete\_flag(msg(x, mbox)) = F$$

As a precondition for **P2** we only know that the message has not been deleted, but information about the state of the mailbox not available.

In contrast to the goal specification $goal_{P1}$, the specification of goals in $goal_{P2}$ specifies no temporary goals, but a *conjunctive* goal:

$$\begin{array}{l}goal_{P2}\colon\ \Diamond[read\_flag(msg(x, mbox)) = T \wedge \\ \qquad\quad delete\_flag(msg(x, mbox)) = T\,]\end{array}$$

The plan **P3** is an example of an *iterative* plan reading all messages from sender *joe* in the mailbox *mbox*. The specification of its preconditions and goals contains universally quantified formulae:

$$\begin{array}{l}pre_{P3}\colon\ open\_flag(mbox) = T \wedge \\ \qquad \forall x\ [sender(msg(x, mbox)) = joe \\ \qquad\quad \rightarrow delete\_flag(msg(x, mbox)) = F\,]\end{array}$$

$$\begin{array}{l}goal_{P3}\colon\ \Diamond\,[\forall x\ [sender(msg(x, mbox)) = joe \\ \qquad\quad \rightarrow read\_flag(msg(x, mbox)) = T \wedge \\ \qquad\qquad delete\_flag(msg(x, mbox)) = T\,]\,]\end{array}$$

Only a very restricted syntactic class of LLP formulae is used for the specification of preconditions and goals. For example, only implicit negation of atomic formulae occurs in implications. Furthermore, atomic formulae are equations assigning constants to terms of a very restricted syntactic structure. The term $msg(x, mbox)$ denotes an arbitrary message in a mailbox. Unary functions like *read_flag* and *delete_flag* represent features of this message. The effects of actions are reflected in changed features.

The plans **P2** and **P3** can be easily adapted in order to obtain the desired plan **P1**:

- **P1** corresponds to the **then**-branch of **P2** when deleting the superfluous *empty_action*.
- **P1** corresponds to the sequential body plan of **P3** when deleting the superfluous iterative control structure and the test on the sender of the message.

Consequently, **P2** and **P3** should both be retrieved from the case library as possible reuse candidates. Furthermore, the retrieval algorithm should differentiate between **P2** and **P3**:

On one hand, both plans **P2** and **P3** are applicable in the initial state specified for **P1** because their preconditions are entailed by pre_{P1}. On the other hand, **P2** is more "similar" to the desired plan than the plan **P3**: it reads and deletes *an arbitrary* message as required in the new case **P1**, while **P3** reads *all* messages from a *particular* sender—an additional condition, which is not required in **P1**.

The identification of **P2** and **P3** as appropriate reusable cases requires *abstraction* from

- specific objects occurring in the specifications,
- temporary subgoal states,
- universally quantified goals.

The effect of actions which reflect in a change of features of a message have to be preserved during the abstraction process.

These requirements are reflected in the definition of the encoding scheme ω, which is used in MRL to map LLP plan specifications to concepts in a terminological logic.

3.2 THE TERMINOLOGICAL LOGIC

The terminological logic $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991] is chosen as a starting point for the terminological part of the representation formalism for case libraries because of its expressiveness and mathematical properties. Concept descriptions in $\mathcal{ALC}$ are built from concepts, intersection, complements and universal role quantifications. The logic possesses a decidable and complete subsumption algorithm which is PSPACE-complete. This means that deciding subsumption in $\mathcal{ALC}$ is intractable. Remember that we required the retrieval algorithm to be efficient, i.e. to run in polynomial time to cope with the scaling-up problem. To obtain polynomial complexity, two solutions can be adopted:

1. Giving up completeness.
2. Restricting the terminological logic.

Giving up completeness in an application system often also implies giving up correctness, because inability to detect existing subsumption relations may lead to incorrect behavior of the system. In particular for case-based systems, the incompleteness of the retrieval algorithm leads to the following problems:

- Existing cases solving the new case may be not found in the case library. This can lead to an undesirable computational overhead in case-based reasoning because the system does not reuse the best available case during problem solving.
- Uncontrolled growth of the case library may occur. Equivalent cases are added to the library because the incomplete subsumption algorithm is unable to recognize the equivalence of indices.

Therefore, the second solution is adopted by restricting concept descriptions to a *normal form* for which a sound, complete and polynomial subsumption algorithm exists. We define a subset of $\mathcal{ALC}$ comprising so-called *admissible concepts* that are consistent concept descriptions in conjunctive normal form. They are only built from *primitive components*, i.e. existential role restrictions of the form $\exists R.C$ and $\exists R.\neg C$ where C is required to be a primitive concept and R is restricted to be a chain of primitive roles. The following subsumption algorithm is defined for admissible concepts $\mathcal{C}_a$:

Definition 2 *$SUBS(u,t) : \mathcal{C}_a^2 \longrightarrow \{true, false\}$*

$SUBS(u,t)$ computes its result using the rules:[1]

$$z \sqsubseteq x, z \sqsubseteq y \quad \rightarrow \quad z \sqsubseteq x \wedge y \tag{1}$$
$$x \sqsubseteq z \quad \rightarrow \quad x \wedge y \sqsubseteq z \tag{2}$$
$$x \sqsubseteq z, y \sqsubseteq z \quad \rightarrow \quad x \vee y \sqsubseteq z \tag{3}$$
$$z \sqsubseteq x \quad \rightarrow \quad z \sqsubseteq x \vee y \tag{4}$$
$$x \quad \sqsubseteq \quad x \tag{5}$$

Theorem 1 *SUBS is sound, complete and decides the subsumption relation in polynomial time for admissible concepts.*

The proof can be found in [Koehler, 1994b].

The expressiveness of admissible concepts is sufficient to adequately represent the mail domain.[2] As an example, consider the LLP formula

$$\Diamond\, read\_flag(msg(x, mbox)) = T$$

The interpretation of this formula is a *message* at a certain *position* in a particular *mailbox* at a certain *world state*, the *read_flag* of which is set to the value

[1] This rule set is equivalent to a sound and complete rule set for lattices given in [Givan and McAllester, 1992] that decides the defined inference relation in polynomial time. Note, that SUBS(u,t) is incomplete for arbitrary concept descriptions in $\mathcal{ALC}$.

[2] This property may not generalize to other application domains, see Section 5.

true. Figure 1 illustrates a subset of the primitive concepts and roles representing the mail domain. Starting with the concept `STATE`, role chains can be composed, which describe the state of a particular message at a particular position in a particular mailbox at a certain world state. Consequently, the admissible concept

$$\exists\, mbox \circ pos \circ mesg \circ read\_flag.T$$

abstracts the LLP example formula.

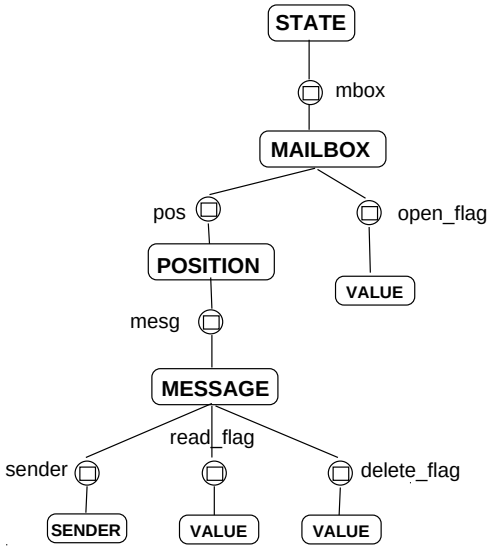

Figure 1: A Subset of the Mail Terminology

3.3 DEFINING THE ENCODING SCHEME

The encoding scheme ω maps LLP plan specifications to indices in $\mathcal{ALC}$ on the basis of the declarative semantics both logics possess. It depends on the source logic LLP as well as on the target logic $\mathcal{ALC}$.

LLP plan specifications are a restricted class of temporal logic formulae containing modal operators. In order to map them to concept descriptions they are translated into first-order predicate logic using the method developed in [Frisch and Scherl, 1991], which has been extended to LLP in [Koehler and Treinen, 1993]. The result of the translation is a formula in a predicate logic with constraints. The constraint theory represents temporal information, e.g. which subgoal has to hold in a particular state. In a next step, the constraint theory is eliminated, preserving the satisfiability of the formula. The elimination of the constraint theory implements a process of temporal abstraction: the temporal information is eliminated from the formula.

The encoding scheme abstracts from specific objects by replacing constants with existentially quantified variables. Furthermore, universal quantification is replaced by the weaker existential quantification. An n-ary function is encoded as an $n+1$-ary relation. Each $n+1$-ary relation is encoded by n binary relations. These abstraction operations are justified by the restricted syntactic structure of terms and formulae.

After the abstraction process has been completed, the conjunctive normal form of preconditions and goals is computed. Of course, the computational effort for this operation grows exponentially with the length of the formulae. But remember that the subsumption algorithm is only complete for concepts in conjunctive normal form. Nevertheless, for pragmatic reasons it is more efficient to compute the normal form only once during the encoding process instead of computing it several times during the classification of an index.

Finally, the remaining set of formulae is syntactically transformed into sets of formulae of the form $\phi_C(x) : \exists\, y\, P(x,y) \wedge Q(y)$. The declarative semantics of terminological logics allow primitive concepts to be seen as unary predicates and primitive roles to be seen as binary predicates. This identification can be extended to arbitrary concept descriptions, i.e. to every concept C a predicate formula $\phi_C(x)$ can be associated. Consequently, a concept $C : \exists P.Q$ corresponds to the formula $\phi_C(x)$. A model of the formula $\exists\, x\, \phi_C(x)$ is a model of the concept C and vice versa. In particular, C is unsatisfiable if and only if $\exists\, x\, \phi_C(x)$ is unsatisfiable [Hollunder *et al.*, 1990].

In the example, the following encoding of preconditions and goals is obtained:[3]

$\omega(pre_{P1})$: $\exists\, mbox \circ open\_flag.T \sqcap$
$\exists\, mbox \circ pos \circ mesg \circ delete\_flag.F$

$\omega(goal_{P1})$: $\exists\, mbox \circ pos \circ mesg \circ read\_flag.T \sqcap$
$\exists\, mbox \circ pos \circ mesg \circ delete\_flag.T$

$\omega(pre_{P2})$: $\exists\, mbox \circ pos \circ mesg \circ delete\_flag.F$

$\omega(goal_{P2})$: $\exists\, mbox \circ pos \circ mesg \circ read\_flag.T \sqcap$
$\exists\, mbox \circ pos \circ mesg \circ delete\_flag.T$

$\omega(pre_{P3})$: $\exists\, mbox \circ open\_flag.T \sqcap$
$[\exists\, mbox \circ pos \circ mesg \circ sender.\neg S \sqcup$
$\exists\, mbox \circ pos \circ mesg \circ delete\_flag.F\,]$

$\omega(goal_{P3})$: $[\exists\, mbox \circ pos \circ mesg \circ sender.\neg S \sqcup$
$\exists\, mbox \circ pos \circ mesg \circ delete\_flag.T\,] \sqcap$
$[\exists\, mbox \circ pos \circ mesg \circ sender.\neg S \sqcup$
$\exists\, mbox \circ pos \circ mesg \circ delete\_flag.T\,]$

3.3.1 Proving the Monotonicity Theorem

To ensure that the retrieval algorithm performs predictably, the monotonicity property has to be proved

[3] TRUE is abbreviated to T, FALSE is abbreviated to F, and SENDER is abbreviated to S.

for the encoding scheme used in MRL.

An old plan solving the old planning problem $\langle pre_{old}, goal_{old}\rangle$ is reused as a solution for a new planning problem $\langle pre_{new}, goal_{new}\rangle$ in MRL if

$$pre_{new} \vdash pre_{old} \text{ and } goal_{old} \vdash goal_{new}$$

can be successfully proved in the logic LLP [Koehler, 1994a]. Therefore, we have to prove the following instance of the monotonicity theorem:

Theorem 2
If $pre_{new} \vdash pre_{old}$ *then* $\omega(pre_{new}) \sqsubseteq \omega(pre_{old})$ *and*
if $goal_{old} \vdash goal_{new}$ *then* $\omega(goal_{old}) \sqsubseteq \omega(goal_{new})$.

The proof of the theorem can be found in [Koehler, 1994b]. The correctness of the encoding scheme used in MRL relies on the syntactic restrictions which are imposed on terms and formulae. Nevertheless, we believe that the general idea to ground the formalization of abstraction on the definition of an encoding scheme is widely applicable.

3.4 FORMALIZING THE RETRIEVAL

The results of the encoding process are the admissible concepts $\omega(pre)$ and $\omega(goal)$ from which the index of a case is obtained as the pair $\langle\omega(pre), \omega(goal)\rangle$. Now, the *retrieval* of a plan from the plan library is formalized as follows:

Given the description of a new case, the index of this case is computed first. Then, this index is classified in the plan library. Two classification operations are available:

- strong classification
- weak classification

Strong classification classifies the new index by computing the required subsumption relations between encodings of preconditions and goals:

$$\omega(pre_{new}) \sqsubseteq \omega(pre_{old}) \quad \underline{\text{and}} \quad \omega(goal_{old}) \sqsubseteq \omega(goal_{new})$$

The result of the classification process determines the position of the new index in the plan library. All indices that are subsumed by the new index are considered as potential reuse candidates. The plans belonging to the subsumed indices are assumed to be applicable in the current initial state and to reach all of the current goals.

In the example, *strong classification* of the new index $\langle\omega(pre_{P1}), \omega(goal_{P1})\rangle$ inserts this index at the position shown in Figure 2. Obviously, $\langle\omega(pre_{P2}), \omega(goal_{P2})\rangle$ is subsumed by the new index. According to Definition 1, the planning problem stored in the plan entry related to the index $\omega(P2)$ is more specific than the new planning problem that has to be solved. The plan **P2** is activated as a possible reuse candidate and sent to the plan modification module of MRL [Koehler, 1994a]. The index of plan **P3** does not meet the criteria required by strong classification, since the subsumption test between the goal concepts fails. This plan is not considered as being similar to the desired plan.

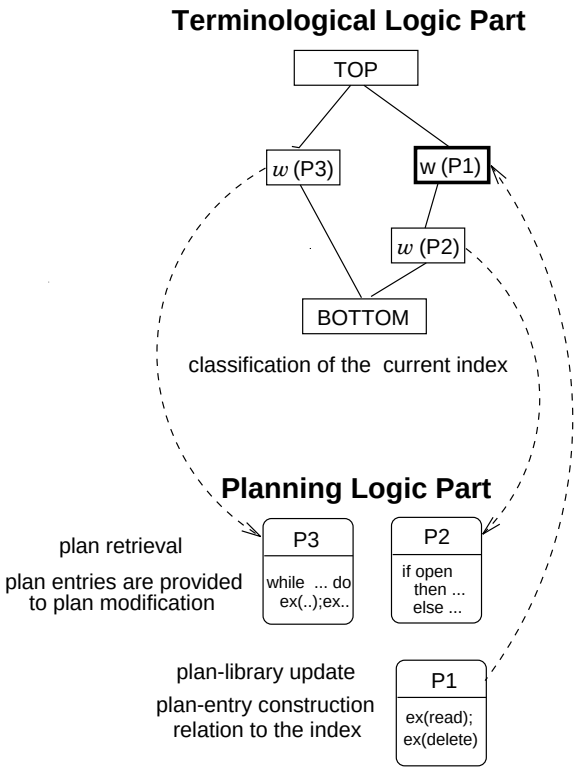

Figure 2: A small Sample Case Library

Weak classification is activated when strong classification *fails* to retrieve a reuse candidate. It is based on a weaker search criterion and can classify according to goals or preconditions:

$$\omega(pre_{new}) \sqsubseteq \omega(pre_{old}) \quad \underline{\text{or}} \quad \omega(goal_{old}) \sqsubseteq \omega(goal_{new})$$

Note, that every case that meets the criteria of strong classification also meets the weaker criterion used by weak classification. In the MRL system, plans are reused if they are applicable in the current initial state. Therefore, weak classification in MRL classifies according to preconditions:

$$\omega(pre_{new}) \sqsubseteq \omega(pre_{old})$$

In the example, *weak classification* retrieves **P2** and **P3**, because the subsumption test on the encodings of preconditions is successful. Nevertheless, plan **P3** is considered as being less appropriate for the solution of **P1** than **P2** according to the weaker search criterion.

Figure 2 illustrates the hybrid representation of the plan library in MRL for the example under consideration. The terminological logic part supports the structuring of the plan library. Retrieval and update are grounded on *classification* by computing the subsumption hierarchy of indices. The planning logic part supports the representation of planning knowledge in plan entries.

3.5 RANKING OF CASES

Strong as well as weak classification can retrieve several appropriate reuse candidates from the case library. Consequently, a *ranking* sequence is needed for the candidates in order to find the best one.

Strong classification determines plans from the plan library that are supposed

- to be applicable in the initial state and
- to achieve at least all of the current goals.

This implies that the candidate set retrieved by strong classification may contain plans which achieve superfluous goals, i.e. goals that are currently unnecessary. Actions achieving these goals have to be eliminated from the reused plan by optimizing it. The ranking of the candidates is therefore grounded on an estimation of the *optimization effort* for each candidate. The ranking heuristic estimates the number of superfluous actions that have to be eliminated from the candidate plan.

Observe that the subsumption hierarchy for indices is defined such that a plan $b1$ achieving more atomic goals than a plan $b2$ is placed closer to the *bottom* concept than $b2$. Consequently, the estimated optimization effort for a plan, the index of which is placed closer to the *bottom* concept, is higher than the estimated optimization effort for a plan, the index of which is immediately subsumed by the current index. Therefore, strong classification only adds a case to the solution set, if its index is immediately subsumed by the current index.

The estimation of the optimization effort proceeds as follows:

- The ranking heuristic compares the goal concept of the current index $\omega(goal_{new})$ with the goal concepts of all immediately subsumed indices $\omega(goal_{old_i})$.
- It computes the number of primitive components, which occur in $\omega(goal_{old_i})$, but not in $\omega(goal_{new})$.
- The case with the smallest number is selected as the best candidate according to the ranking heuristic.

The heuristic estimates the number of atomic subgoals that are achieved by a candidate plan but that are not required in the current plan specification. It assumes that this number reflects the minimal number of primitive actions in the candidate plan that have to be eliminated. Therefore, the plan with the smallest number is selected as the best reuse candidate and sent to the plan modification module. If several candidates receive the same ranking value, one of them is selected arbitrarily.

Definition 3 *Let $C_{old_1}, \ldots, C_{old_n}$ be the set of candidates retrieved by strong classification of $\omega(C_{new})$. The goal concepts occurring in the indices of the candidates are $\omega(goal_{old_1}), \ldots, \omega(goal_{old_n})$, the goal concept occurring in the current index is $\omega(goal_{new})$. The set of primitive components that occurs in a concept c is denoted by $\mathsf{PK}[c]$, while their number is denoted by $\mathsf{N}[c]$.*

The optimization effort for each candidate is defined as

$$OPT_{\omega(goal_{old_i})} = \mathsf{N}\Big[\mathsf{PK}[\omega(goal_{old_i})] \setminus \mathsf{PK}[\omega(goal_{new})]\Big]$$

The ranking heuristic $\mathcal{H}_{OPT}$ selects the candidate with the smallest optimization effort:

$$\mathcal{H}_{OPT} = \Big\{ C_{old_i} | OPT_{\omega(goal_{old_i})} = min\Big(OPT_{\omega(goal_{old_1})}, \ldots, OPT_{\omega(goal_{old_n})}\Big) \Big\}$$

Weak classification determines plans from the plan library that are only supposed to be applicable in the initial state.

The goal concepts of the candidate plans can be related to the goal concept of the current case in two ways:

1. $\omega(goal_{new}) \sqsubseteq \omega(goal_{old})$
   This means that we can expect the candidate plan to achieve only a subset of the goals required in the current case.
2. $\omega(goal_{new}) \not\sqsubseteq \omega(goal_{old})$ and
   $\omega(goal_{old}) \not\sqsubseteq \omega(goal_{new})$
   No subsumption relation holds for the goal concepts of the candidate and the current case. We have to expect that the candidate achieves other goals than those required in the current plan specification.

Therefore, the ranking heuristic for candidates retrieved by weak classification relies on the following assumptions:

- Every candidate is applicable in the current initial state.
- No candidate achieves all of the current goals, i.e. every candidate has to be modified.

Consequently, the heuristic estimates the *modification effort* for each candidate as follows:

- The ranking heuristic compares the goal concept of the current index $\omega(goal_{new})$ with the goal concepts $\omega(goal_{old_i})$ of all indices occurring in the solution set.
- It computes the intersection of the concepts, i.e. the number of primitive components occurring in $\omega(goal_{new})$ as well as in $\omega(goal_{old_i})$.
- This number measures the modification effort by an estimation of the number of current atomic goals that are achieved by each candidate.

The candidate with the biggest number is selected as being the best reuse candidate, because it is assigned the highest "success rate" and therefore its modification effort is estimated as being minimal. Furthermore, the ranking heuristic verifies whether the ranking value of the best candidate exceeds a lower bound: it requires that at least half of the primitive components from $\omega(goal_{new})$ must be contained in $\omega(goal_{old_i})$. If this condition is satisfied, the ranking heuristic assumes that the best candidate achieves at least half of the current atomic goals.[4]

Definition 4 *Let $C_{old_1}, \ldots, C_{old_n}$ be the set of candidates retrieved by weak classification of $\omega(C_{new})$. The goal concepts occurring in the indices of the candidates are $\omega(goal_{old_1}), \ldots, \omega(goal_{old_n})$, the goal concept occurring in the current index is $\omega(goal_{new})$. The set of primitive components that occurs in a concept c is denoted by $\mathsf{PK}[c]$, while their number is denoted by $\mathsf{N}[c]$.*

The estimated success rate for each candidate is defined as:

$$MOD_{\omega(goal_{old_i})} = \mathsf{N}\Big[\mathsf{PK}[\omega(goal_{old_i})] \cap \mathsf{PK}[\omega(goal_{new})]\Big]$$

The ranking heuristic $\mathcal{H}_{MOD}$ selects the candidate with the biggest success rate that exceeds the lower bound:

$$\mathcal{H}_{MOD} = \Big\{ C_{old_i} \mid MOD_{\omega(goal_{old_i})} = max\Big(MOD_{\omega(goal_{old_1})}, \ldots, MOD_{\omega(goal_{old_n})}\Big) \text{ and } MOD_{\omega(goal_{old_i})} \geq \frac{\mathsf{N}[\omega(goal_{new})]}{2} \Big\}$$

If no candidate receives a ranking value which exceeds the lower bound, all candidates are rejected because their modification effort is too costly. In this situation, case-based planning reports a failure and planning from scratch with the PHI planner is activated.

The ranking heuristics guide the interaction between case-based planning and plan generation, see Figure 3. Plan generation is activated when

- no candidate can be retrieved from the library,
- the modification effort is estimated as being too costly for all potential candidates.

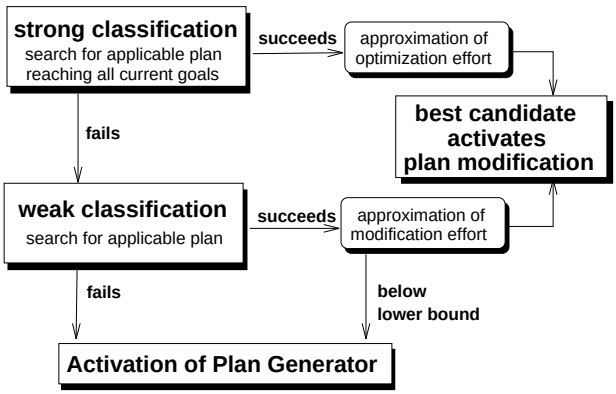

Figure 3: Heuristic Guidance of Case-based Planning

The *update* of the plan library is activated when

- no reusable plan is found and planning from scratch is performed,
- the retrieved plan has to be optimized or modified.

During the update of the plan library a new plan entry is built. Three sources of information are available: the formal plan specification C_{new}, the generated or modified plan S_{new} and the proofs performed during deductive plan generation and plan modification [Koehler, 1994a]. The plan entry is built out of C_{new}, S_{new} and information that is extracted from the proofs. It is related to its index $\omega(P_{new})$ that was already computed and classified during the retrieval process. The index determines the position of the new plan entry in the plan library. It is now available for a subsequent case-based planning process.

4 IMPLEMENTATION

The system MRL has been implemented as an integrated part of the PHI system in SICSTUS Prolog. The plan library can be *static* as well as *dynamic*:

A static library comprises user-predefined typical plans. The system retrieves these plans for reuse, but does not add new plans to the library. A dynamic plan library grows during the lifetime of the system. MRL starts with an empty library and incrementally adds new plan entries to it. The system thus automatically builds a taxonomy of abstract descriptions of typical planning problems that occur in the application domain.

The application of terminological logics leads to remarkable properties of the system:

[4] The definition of an appropriate lower bound may differ for different case-based planning systems.

The mapping of specific planning problems to abstract classes helps to keep the plan library small. Only one representative for each class is added to the plan library. Instances of planning problems which belong to the same class can be solved by instantiation or easy modification of the retrieved candidate plan. Furthermore, the implementation of the representational shift from specific planning problems to abstract problem classes with the help of the encoding scheme requires only marginal computational costs.

The polynomial complexity of the subsumption algorithms leads to an efficient retrieval of candidate plans in polynomial time, cf. [Koehler, 1994b].

The completeness of the subsumption algorithm ensures that existing solutions are found in the plan library. This leads to efficiency gains of the case-based planner compared to the generative planner because the system can reuse any solution that exists in the plan library.

5 RELATED WORK

Recently, the representation of plans based on terminological knowledge-representation systems has led to several approaches, which extend terminological logics with new application-oriented representational primitives for the representation of actions and plans.

One such an extension is the system RAT [Heinsohn *et al.*, 1991] which is based on $\mathcal{KRIS}$ [Baader *et al.*, 1992]. RAT is able to implement reasoning about plans by inferences in the underlying terminological logic. The system simulates the execution of plans, verifies the applicability of plans in particular situations and solves tasks of temporal projection.

An application of terminological logics to tasks of plan recognition is developed in T-REX [Weida and Litman, 1994]. Plans in T-REX may contain conditions and iterations as well as non-determinism in the form of disjunctive actions.

More complex application domains may require the integration of more expressive terminological logics into the hybrid representation formalism for case libraries. A future direction of work is the integration of stochastic approaches and the parallelization of the search. A successful application of a probabilistic method for NP-complete inference problems is described in [Selman *et al.*, 1992]. The usefulness of non-systematic search strategies in planning is demonstrated in [Langley, 1992; Minton *et al.*, 1992].

6 CONCLUSION

We have presented an application of terminological logics as a kind of query language in case-based reasoning. Indices are built from concept descriptions. The retrieval and update operations working on case libraries are formalized as classification operations over the taxonomy of indices.

An example taken from the field of case-based planning demonstrates the applicability of the theoretical framework. The behavior of the case-based planner becomes predictable and theoretical properties like the correctness, completeness and efficiency of the retrieval algorithm can be proved.

Acknowledgements

I am indebted to Wolfgang Wahlster for his advice and support. I wish to thank Hans-Jürgen Profitlich who helped me test the practical feasibility of the approach with the development of a prototypical plan library in RAT, and Bernhard Nebel and Hans-Jürgen Ohlbach for fruitful discussions regarding the theoretical properties of the formalism. Werner Nutt and the anonymous referees made helpful comments on a draft version of this paper.

References

J.A. Allen, R. Fikes, and E. Sandewall, editors. *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, April 1991. Morgan Kaufmann.

F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems, or making KRIS get a move on. In Nebel et al. 1992, pages 270–281.

M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul. PHI - a logic-based tool for intelligent help systems. In IJCAI-93, pages 460–466.

S. Biundo and D. Dengler. The logical language for planning LLP. Research Report, German Research Center for Artificial Intelligence, 1994.

S. Biundo, D. Dengler, and J. Koehler. Deductive planning and plan reuse in a command language environment. In Neumann 1992, pages 628–632.

R. Brachman. Structured inheritance networks. In W. Woods and R. Brachman, editors, *Research in Natural Language Understanding*, pages 36–78. Bolt, Beranek, and Newman Inc., Cambridge Mass., 1978.

R. Brachmann and H. Levesque. The tractability of subsumption in frame based description languages. *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, pages 34–37, Austin, TX, 1984. MIT Press.

CBR-91 *Proceedings of the 3rd Case-Based Reasoning Workshop*, Washington, D.C., 1991. Morgan Kaufman, San Mateo.

CBR-93 *Proceedings of the AAAI-93 Workshop on Case-Based Reasoning*, number WS-93-01 in AAAI Technical Report, Washington, D.C., 1993. AAAI Press, Menlo Park.

E.A. Feigenbaum. The simulation of natural learning behavior. In E.A. Feigenbaum and J. Feldman, editors, *Computers and Thought*. Mc Graw-Hill, New York, 1963.

A. M. Frisch and R. B. Scherl. A general framework for modal deduction. In Allen et al. 1992, pages 196–207.

R. Givan and D. McAllester. New results on local inference relations. In Nebel et al. 1992, pages 403–412.

J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. Integration of action representation in terminological logics. In C. Peltason, K. Luck, and C. Kindermann, editors, *Proceedings of the Terminological Logic Users Workshop*. KIT–Report 95, TU Berlin, Germany, 1991.

B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In L.C. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden, August 1990. Clays Ltd, England.

IJCAI-93 *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, August 1993. Morgan Kaufmann.

H. Kautz and B. Selman. Planning as satisfiability. In Neumann 1992, pages 359–363.

J. Koehler and R. Treinen. Constraint deduction in an interval-based temporal logic. In *Working Notes of the AAAI Symposium on Automated Deduction in Nonstandard Logics*. AAAI Press, Menlo Park, 1993.

J. Koehler. Flexible plan reuse in a formal framework. In C. Bäckström and E. Sandewall, editors, *Current Trends in AI Planning*. pages 171–184. IOS Press, Amsterdam, Washington, Tokyo, 1994.

J. Koehler. *Reuse of Plans in Deductive Planning Systems*. PhD thesis, University of Saarland, 1994. in German.

J. Kolodner. *Case-Based Reasoning*. Morgan Kaufman, 1993.

P. Langley. Systematic and nonsystematic search strategies. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, pages 145–152, Washington, D.C., 1992. Morgan Kaufmann, San Mateo.

S. Minton, M. Drummond, J. Bresina, and A. Philips. Total order vs. partial order planning: Factors influencing performance. In Nebel et al. 1992, pages 83–92.

B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In IJCAI-93, pages 1436–1441.

B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. Research Report RR-93-33, German Research Center for Artificial Intelligence (DFKI), 1993.

B. Nebel, W. Swartout, and C. Rich, editors. *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, October 1992. Morgan Kaufmann.

B. Neumann, editor. *Proceedings of the 10th European Conference on Artificial Intelligence*, Vienna, Austria, August 1992. John Wiley & Sons.

C.K. Riesbeck and R.C. Schank. *Inside Case-based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 440–446, San Jose, CA, July 1992. MIT Press.

S. Slade. Case-based reasoning: A research paradigm. *The AI Magazine*, 12(1):43–55, 1989.

R. Weida and D. Litman. Subsumption and recognition of heterogeneous constraint networks. In *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*, 1994. to appear.