Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-94-13

Planning from Second Principles —A Logic-based Approach

Jana Koehler

June1994

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

Planning from Second Principles—A Logic-based Approach

Jana Koehler

DFKI-RR-94-13

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-9000 8).

ISSN 0946-008X

Planning from Second Principles - A Logic-based Approach -

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

Abstract

In this paper, a logical formalization of planning from second principles is proposed, which relies on a systematic decomposition of the planning process. Deductive inference processes with clearly defined semantics formalize planning from second principles.

Plan modification is based on a deductive approach which yields provably correct modified plans.

Reusable plans are retrieved from a dynamically created plan library using a description logic as a query language to the library.

Apart from sequential plans, this approach enables a planner to efficiently reuse and modify plans containing control structures like conditionals and iterations.

Contents

1 Introduction

Planning from first principles generates plans from "scratch", e.g., the planner searches for a set of actions (a plan) that achieves the desired goal with respect to specified preconditions. A serious limitation of first-principles planners is the constancy of the planning process over time: If a planner receives the same planning problem, it will repeat exactly the same planning operations. In other words, it is unable to benefit from experience that can be drawn from previous planning processes.

Approaches to *planning from second principles* try to overcome this limitation of planning from scratch by flexible reuse and modification of plans. In cases of execution failures, where a plan has to be revised in the light of new information, and changes of plan specifications, where a plan has to meet new requirements, modification of the existing plan seems more reasonable than generating a new one.

The current state of the art comprises a variety of approaches tackling the problems from a cognitive point of view (cf. [Kolodner, 1993] for a summary of approaches) or in the framework of STRIPS-based planning, cf. [Kambhampati and Hendler, 1992; Hanks and Weld, 1992a; Veloso, 1992a].

Using a formal framework, we present a logic-based approach to planning from second principles, which makes no commitments to particular planning formalisms and application domains.

Plan modification is based on deductive inference processes that yield provably correct modified plans. As a new issue in plan modification we discuss the reuse and refitting of control structures occurring in plans, like case analyses and iterations, which introduce qualitatively new problems.

As for the plan library we propose a hybrid knowledge representation formalism linking the planning logic with a description logic. In this approach, description logics are used as a kind of query language to the plan library. This leads to well-defined abstraction, retrieval and update procedures possessing interesting theoretical and practical properties. In particular, we demonstrate that the bottleneck of plan retrieval [Nebel and Koehler, 1993b] can be overcome by developing efficient approximation algorithms that are guaranteed to find existing solutions to current planning problems in the plan library.

Summarizing, the properties of this new approach are:

- Second-principles planning is addressed in a strictly logic-based way.
- The planner is based on a general unified formal framework. It covers the modification of plans and the representation of the plan library, including retrieval and update operations.

- Besides simple sequential plans, this approach enables a second-principles planner to flexibly reuse plans containing control structures like case analysis and iteration.
- The formal framework allows us to prove important properties like the correctness and completeness of the underlying inference procedures.

The paper is organized as follows: The foundation of the logic-based approach is a four-phase model, described in Section 2. It supports a temporal view as well as a task-specific view on the second-principles planning process. In Section 3 the formal framework of planning from second principles is presented. It provides the theoretical basis for the system MRL that is described in the main part of this paper. MRL extends the generative deductive planner PHI, which is briefly introduced in Section 4, with the ability to reuse and modify plans that are stored in a dynamically updated plan library.

Sections 5 and 7 are devoted to the inference procedures working on the plan library, while in Section 6 the deductive approach to plan modification is presented. Finally, in Section 8 we propose a systematic categorization of the various principles and design decisions underlying second-principles planners, and summarize the main properties of the MRL planner in the light of this categorization.

2 A Four-Phase Model

Second-principles planners work in a basic cycle of *problem input—activation of previous plans—adaptation*, cf. [Riesbeck and Schank, 1989]. The problem of plan activation addresses questions of how to represent and store previous plans and knowledge extracted from planning processes including learning and abstraction as well as the problem of how to retrieve relevant information from a plan library, including organization, indexing and search. The problem of adapting an old plan to a new planning problem comprises a *matching* or *test* phase where plans are matched against current planning problems or executed in simulated environments and a subsequent *refitting/adaptation/repair* phase where the plan is modified in accordance with the result of the matching or test.

We distinguish four phases in a second-principles planning process:

Phase I achieves retrieval by a process called *plan determination.*

(I) Plan Determination: The description of a current planning problem, i.e., the current plan specification is the input to the plan-determination

phase. From this specification, an index has to be computed, which represents the search key to the plan library. The plan library contains a collection of *plan entries* that are extracted from previously solved planning problems. A plan entry provides comprehensive information about a planning problem and its solution, e.g., the specification of the problem describing initial and goal states, the plan which was generated as a solution for it, and information that is extracted from the plan generation process. Each plan entry possesses an index which determines its position in the hierarchically organized plan library.

The current search key is matched against the indices of the stored plan entries. Based on the result of the matching a set of reuse candidates is determined. Ranking heuristics are applied in order to determine the best candidate.

Phase II compares of the current plan specification with the reused plan specification. It is based on a deductive inference process called *plan interpretation.*

(II) Plan Interpretation: Plan interpretation attempts to prove that the current plan specification is a *logical instance* of the reused plan specification by proving relations between preconditions and goals. A successful proof attempt means that the reused plan specification is sufficient for the current one, i.e., solving the old planning problem will solve the current planning problem.

In contrast to a purely syntactic matching, knowledge about regularities in the application domain can be applied during the proof. The result of the plan-interpretation phase is a plan skeleton in the case of a proof failure. An instance of the library plan that solves the current planning problem is obtained when the proof is successful.

Phase III completes the plan skeleton to a *correct plan* with the help of an interleaved process of plan verification and generative planning called *plan refitting.*

(III) Plan Refitting: A plan skeleton provides an entry point into the search space of possible plans. It represents an incomplete solution to the current planning problem, because it may contain "placeholders" for subplans achieving open subgoals, which the reused plan is unable to achieve. The plan skeleton keeps any actions of the reused plan that were determined as reusable during plan interpretation and in which variables are appropriately instantiated with object parameters taken from the current plan specification.

The planning process terminates with a *plan-library update* in phase IV.

(IV) Plan-Library Update: A new plan entry is constructed from three sources of information: the current plan specification, the plan which was generated by modifying an existing plan, and information that is extracted from the proof tree which was constructed as a result of the completion of the plan skeleton. The plan entry is related to the current index that was computed in phase I. The modified plan is now available to subsequent planning processes.

Figure 1 shows the architecture of the MRL system. The system comprises four modules, each of which implements a phase occurring during second-principles planning.

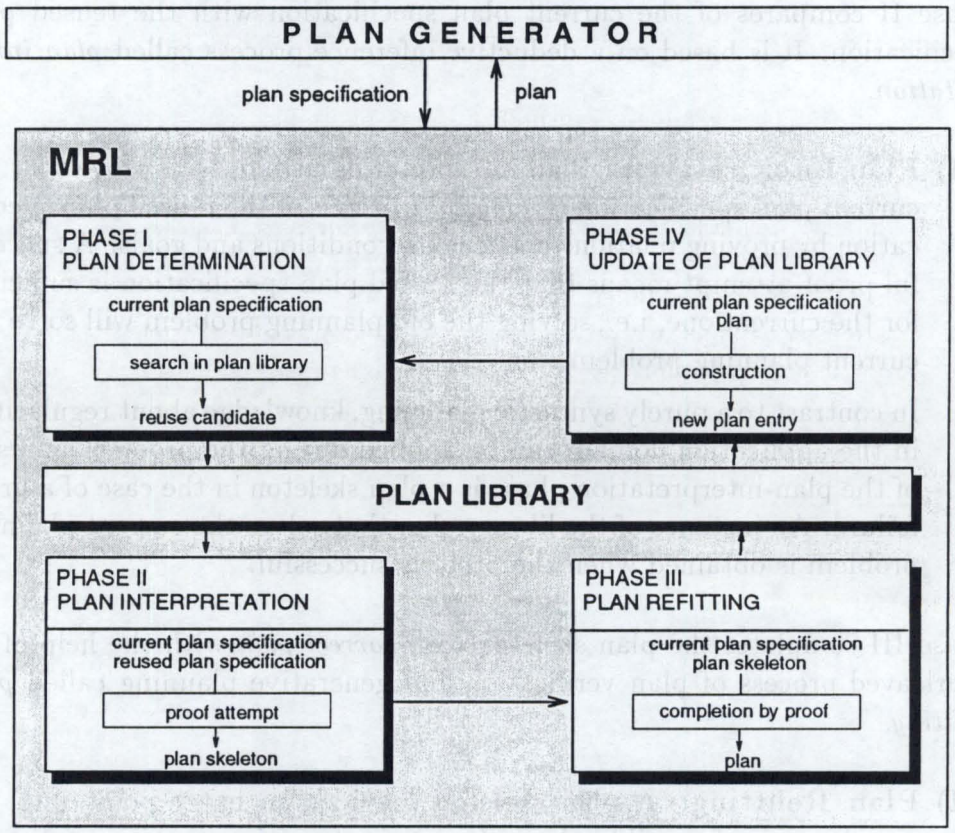

Figure 1: Architecture of the MRL system

The four phase model describes a temporal view on the reuse process. The phases I to III are necessary to generate a plan by reusing an existing one. They are also distinguished by other authors who sometimes denote them

as *retrieval - matching - adaptation* phases, cf. [Hanks and Weld, 1992a]. The fourth phase comprises the maintenance of the plan library. The phases which perform similar tasks are grouped together, so that the reuse process can be formalized. Operations on the plan library provide the basis for phases I and IV, while plan interpretation and refitting (phases II and III) work on plan specifications and are summarized as *plan modification.*

3 The Formal Framework

The formal framework assumes that planning problems are specified in a formal way. We presuppose some kind of logical planning formalism with planning problems given as formal plan specification formulae in the underlying planning logic. In particular, we develop our framework in the general setting of *deductive planning* as introduced in [Green, 1969]. Deductive planning generates plans by performing constructive proofs of formal plan specifications, i.e., "to construct a plan that will meet a specified condition, one proves the existence of a state in which the condition is true", cf. [Manna and Waldinger, 1987b], page 14. Usually, this requires us to prove constructively plan specifications of the form

$$\forall s_0 \, \forall a \, \exists z \, Q[s_0, a, z]$$

where s_0 denotes the initial state, a is an argument or input parameter, and z is a planvariable representing the plan term that has to be constructed, cf. [Manna and Waldinger, 1987b]. Recently, an approach to deductive planning has been introduced in [Biundo *et al.*, 1992; Bauer *et al.*, 1993] with plans represented as formulae. Plan specifications are universally quantified formulae of the form $\mathsf{Plan} \wedge \mathit{pre} \rightarrow \mathit{goal}$, cf. Section 4. In the following, we develop the formal framework for planning from second principles using this latter form of plan specifications.

In general, a plan specification comprises the description of

- an *initial state*, i.e., the preconditions, *pre*, that can be assumed to hold,
- a *goal state*, *goal*, which has to be achieved by executing the plan.

Planning from first principles receives a current plan specification S_{new} and tries to find a $\mathsf{Plan}_{\mathsf{new}}$ by "inspecting" the set of available actions. Planning from second principles tries to find a $\mathsf{Plan}_{\mathsf{new}}$ by adapting a $\mathsf{Plan}_{\mathsf{old}}$ solving an old plan specification S_{old} in such a way that it solves S_{new}.

In order to formalize the retrieval of candidate plans from the plan library we have to answer the question

Question 1: "How to find a $\mathsf{Plan}_{\mathsf{old}}$ possibly solving S_{new}?"

while the formalization of plan modification requires an answer to the question

Question 2: "Does $\mathsf{Plan}_{\mathsf{old}}$ solve S_{new}?"

3.1 Plan Modification

The answer to Question 2 is found with the help of Definition 1:

Definition 1 $\mathsf{Plan}_{\mathsf{old}}$ *solves* S_{new} *if and only if* $Ax \vdash S_{old} \rightarrow S_{new}$.

This means, S_{old} is sufficient for S_{new} under the axiomatization Ax of the considered application domain, i.e., $Ax, S_{old} \models S_{new}$. In other words, S_{new} specifies a logical instance of S_{old}. Thus, solving S_{old} is sufficient for solving S_{new} and consequently, an instance of $\mathsf{Plan}_{\mathsf{old}}$ will solve S_{new}.
Since plan specifications contain formal descriptions of initial and goal states, we can show that $Ax \vdash S_{old} \rightarrow S_{new}$ holds by proving sufficient relations between preconditions and goals according to Theorem 1:

Theorem 1 $Ax \vdash S_{old} \rightarrow S_{new}$ *holds if*

$$Ax \vdash pre_{new} \rightarrow pre_{old} \text{ and } Ax \vdash goal_{old} \rightarrow goal_{new}.$$

This means, we have to prove that the preconditions required by the old plan are satisfied in the current initial state and that the goals achieved by the old plan are sufficient for the currently required goals. If these relationships between initial and goal state specifications hold, we know that

- $\mathsf{Plan}_{\mathsf{old}}$ is applicable in pre_{new} and
- $\mathsf{Plan}_{\mathsf{old}}$ achieves at least all of the goals required in $goal_{new}$.

Proof:
The validity of Theorem 1 is obvious. Assuming that plan specification formulae are of the form $\mathsf{Plan} \wedge pre \rightarrow goal$ we ground plan modification on a proof of a formula of the form

(1) $[\mathsf{Plan}_{\mathsf{old}} \wedge pre_{old} \rightarrow goal_{old}] \rightarrow [\mathsf{Plan}_{\mathsf{new}} \wedge pre_{new} \rightarrow goal_{new}]$

under the domain axiomatization. Equivalent transformations of this formula lead to a conjunction of the following three formulae, the proof of which is sufficient for the validity of formula (1)

(1a) $\boxed{\mathsf{Plan}_{new}} \wedge pre_{new} \rightarrow \boxed{\mathsf{Plan}_{old}} \vee goal_{new}$

(1b) $\mathsf{Plan}_{new} \wedge \boxed{pre_{new}} \rightarrow \boxed{pre_{old}} \vee goal_{new}$

(1c) $\mathsf{Plan}_{new} \wedge pre_{new} \wedge \boxed{goal_{old}} \rightarrow \boxed{goal_{new}}$

A closer look at these formulae reveals that they lead on one the hand to a valid proof if $pre_{new} \rightarrow goal_{new}$ can be proved, i.e., the current plan specification is a tautology. But in most cases this will not be case and it is not the aim of the proof to act as a tautology checker. On the other hand, we can prove relationships between subformulae from S_{new} and S_{old}. Thus, we obtain three subproofs that are sufficient for the validity of the relation between the two plan specifications $S_{old} \rightarrow S_{new}$:

$$\begin{array}{lrcl} \text{(I)} & \mathsf{Plan}_{\mathsf{new}} & \Rightarrow & \mathsf{Plan}_{\mathsf{old}} \\ \text{(II)} & pre_{new} & \Rightarrow & pre_{old} \\ \text{(III)} & goal_{old} & \Rightarrow & goal_{new} \end{array}$$

The first subproof reflects the aim of second-principles planning: $\mathsf{Plan}_{\mathsf{old}}$ is identified with $\mathsf{Plan}_{\mathsf{new}}$, i.e., the planvariable Plan_{new}, which represents the plan that has to be found, is instantiated with the plan $\mathsf{Plan}_{\mathsf{old}}$ taken from the plan library.

The reader may note that we made no assumption about a particular planning logic or planning calculus. Furthermore, a similar theorem can be obtained when syntactically different plan specifications are used as for example in [Green, 1969; Kowalski, 1979; Manna and Waldinger, 1987b; Manna and Waldinger, 1987a]. In this case, planvariables representing *plan terms* occur as additional arguments in the goal-state specifications. ■

Thus, plan modification is based on attempting to prove relations between preconditions and goals. If the proofs are successful, an instance of $\mathsf{Plan}_{\mathsf{old}}$ will solve S_{new}. This instance is obtained by applying substitutions to $\mathsf{Plan}_{\mathsf{old}}$ that were computed during the proof. If the proof attempt fails, refitting information can be extracted from it, cf. Section 6.

3.2 The Plan Library

In principle, the inference procedures working on the plan library can be formalized in the same way as plan modification. A plan solving the current planning problem can be found by proving sufficient conditions between

preconditions and goals. But obviously, this is too restrictive because such a search process can only retrieve solutions, i.e., plans that are applicable in the current initial state and achieve at least all of the current goals. But an appropriate reuse candidate is a plan that can be properly instantiated to obtain the desired solution or that can be "easily" revised. Therefore, we have to ground the retrieval process on a "relaxation" of these conditions.

Usually, such a "relaxation" is performed on the inference relation, i.e., so-called *partial matches* are computed, cf. [Kolodner, 1993]. The disadvantage of such an approach is that we give up clear semantics of the inference relation and therefore may lose soundness as well. In order to avoid this, we propose an alternative approach by defining an *encoding scheme* ω mapping formal plan specifications to abstract indices.

The encoding scheme formalizes an *abstraction process*: A given plan specification (formula) is mapped to an abstract index (formula) reflecting the main features of the underlying planning problem. Furthermore, we want the abstraction process to be well-defined, i.e., if a plan from the library provides a solution to the current planning problem this plan must be in the retrieval set. The advantage of such a property is obvious. The efficiency of planning from second principles depends on the reuse of existing solutions. Whenever a planning problem can be solved by directly reusing a plan from the library, the system should be able to find this plan in order to minimize the plan-modification effort. This property of the encoding scheme ω is formally stated in Condition 1:

Condition 1 *If* $S_{old} \rightarrow S_{new}$ *then* $\omega(S_{old}) \rightarrow \omega(S_{new})$.

Condition 1 gives a *monotonicity property* of ω and has to be proved for each particular encoding scheme used in a second-principles planner. It expresses that an existing subset relationship between the models M of S_{old} and S_{new} is preserved as a subset relationship between the models of the indices $\omega(S_{old})$ and $\omega(S_{new})$:

$$\text{If } M[P_{old}] \subseteq M[P_{new}] \text{ then } M[\omega(P_{old})] \subseteq M[\omega(P_{new})].$$

The *monotonicity property* of the encoding scheme guarantees that an *existing solution* can be found by searching the plan library along the $\rightarrow$ dimension between indices. Note that the inverse of the monotonicity property does not hold in general. A plan retrieved from the library, the index of which entails the new index, will not, with certainty, provide a solution to the new planning problem. This reflects *reasoning by approximation*. The retrieval algorithm approximates the relationship between the plan specifications when it compares the indices of the plan entries. 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 plan specifications and plans,
- the representation formalism for indices,
- the application domain.

In Section 5, we illustrate the definition of an encoding scheme used in the second-principles planning system MRL. The planning formalism used by MRL is a *temporal logic*. The representation formalism for the indices is a *description logic*, i.e., indices are represented as concepts in a KL-ONE like concept language. With that, the indexing of the plan library is grounded on the *subsumption relation* ($\sqsubseteq$) between indices and a *classifier* is used to retrieve candidate plans. This overcomes the problem of defining *partial matches* between cases, the semantics of which remains often unclear. Furthermore, theoretical properties of the retrieval like its soundness, completeness and runtime complexity can be proved.

Figure 2 summarizes the formal framework. It bases planning from second principles on deductive inference processes. Plan modification is formalized by proving sufficient conditions between preconditions and goals in the underlying planning logic. A formalization of the inference procedures working on the plan library is obtained by computing their approximation in a description logic.

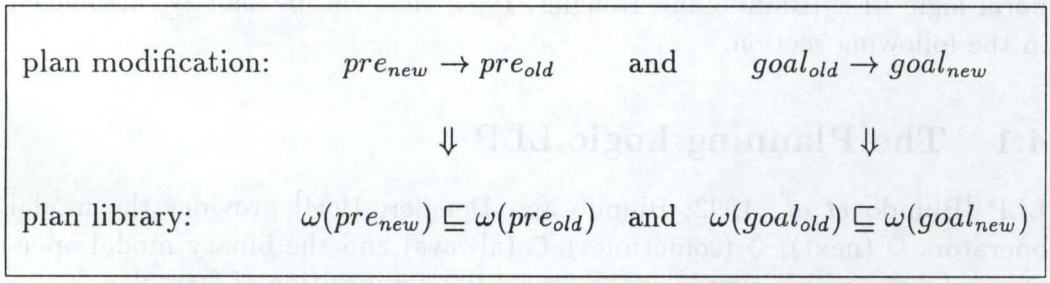

Figure 2: The logical framework for planning from second principles

The idea of exploiting relationships between preconditions and goals can be found in other approaches as well, but they are restricted to a syntactical check of these relations. Hanks and Weld [Hanks and Weld, 1992a] write that "retrieval takes the problem's *initial* and *goal conditions* and finds in the plan library a plan that has worked under circumstances *similar* to those posed by the current problem". The basic approach described by Hammond [Hammond, 1990] is "to find a past plan in memory that satisfies as many

of the *most important goals* as possible". Plan modification as formalized by Kambhampati and Hendler [Kambhampati and Hendler, 1992] relies on marking "the *differences* between the *initial* and *goal state specifications*".

In the remaining part of the paper we show how this formal framework serves as a theoretical foundation for the implemented second-principles planner MRL. The formal framework allows us to implement well-defined inference processes. Thus, the behavior of the system becomes predictable, and theoretical properties like soundness, completeness, and efficiency of the inference procedures can be proved.

4 Second-Principles Planning with MRL

The system MRL[1] has been developed as an integrated part of the system PHI by extending the PHI planner with the ability to reuse and modify complex plans. PHI is a logic-based tool for intelligent help systems which integrates plan generation and plan recognition components [Biundo *et al.*, 1992; Bauer *et al.*, 1993]. Planner and recognizer work in close mutual cooperation, e.g., generated plans can serve as hypotheses for the recognition process [Bauer and Paul, 1994].

A prototypical application of PHI is the UNIX *mail domain* where objects like *messages* and *mailboxes* are manipulated by actions like *read*, *delete*, and *save*.

The logical framework of PHI, which is also used by MRL, is the modal temporal logic LLP [Biundo and Dengler, 1994] that will be shortly introduced in the following section.

4.1 The Planning Logic LLP

LLP [Biundo *et al.*, 1992; Biundo and Dengler, 1994] provides the modal operators $\circ$ (next), $\Diamond$ (sometimes), $\Box$ (always) and the binary modal operator ; (chop) which expresses the sequential composition of formulae. As in programming logics, *local variables* the value of which may vary from state to state are available. Furthermore, control structures like iterations and conditionals can be defined as operators in LLP.

Plans are represented by a certain class of LLP formulae. They may contain basic actions which are expressed by the *execute* predicate *ex*, the argument of which is an action term, the *chop* operator for the sequential composition of plans and actions, and control structures like **if-then-else** and **while** that can be expressed in this logic.

[1] MRL stands for *M*odification and *R*euse in *L*ogic.

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. Changes of state caused by executing 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 $mbox$[2] reads

$$\forall x\,[\,open\_flag(mbox) = T \wedge delete\_flag(msg(x, mbox)) = F \wedge ex(delete(x, mbox)) \rightarrow \circ\, delete\_flag((msg(x, mbox)) = T\,]$$

The state of a mailbox is represented with the help of *flags*. As a precondition, the *delete*-command requires 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. Thus, *plan specifications* are LLP formulae of form

$$\mathsf{Plan} \wedge preconditions \rightarrow goals,$$

i.e., if the Plan is carried out in the initial state where the *preconditions* hold then a state will be achieved where the *goals* hold. Plan is a planvariable, i.e., a non-logical variable, representing the plan formula that has to be generated by performing constructive proofs of the plan specification in a sequent calculus which was developed for LLP. During the proof, the planvariable Plan is replaced by a plan formula satisfying the specification. The proofs are guided by tactics that can be described in a tactic language provided by the system, an idea which was borrowed from the field of *tactical theorem proving* [Constable, 1986; Heisel *et al.*, 1990; Paulson, 1990]. The use of tactics supports the declarative representation of control knowledge and makes deductive planning more efficient. The search space considered during the proof can be kept to a manageable size and only those deduction steps which appear to be the most promising are performed.

Let us consider three specifications of example plans that will be used throughout this paper. The first specification S_{P1} specifies a plan **P1** for the planning

[2]Constants begin with capital letters, while variables are written in lower case.

problem "read and delete a message m in the mailbox *mybox*". As preconditions, we assume that the mailbox *mybox* has already been opened and that the message m has not yet been deleted.

$$\mathbf{S_{P1}}: \quad \begin{array}{l} \mathsf{Plan}_{P1} \wedge \\ open\_flag(mybox) = T \wedge delete\_flag(msg(m, mybox)) = F \\ \rightarrow \Diamond\,[\,read\_flag(msg(m, mybox)) = T \wedge \\ \qquad \Diamond\,[\,delete\_flag(msg(m, mybox)) = T\,]\,] \end{array}$$

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 second specification S_{P2} specifies an example of a *conditional* plan **P2** which reads and deletes a message x in a mailbox *mbox*.

$$\mathbf{S_{P2}}: \quad \begin{array}{l} \mathsf{Plan}_{P2} \wedge delete\_flag(msg(x, mbox)) = F \\ \rightarrow \Diamond\,[\,read\_flag(msg(x, mbox)) = T \wedge \\ \qquad delete\_flag(msg(x, mbox)) = T\,] \end{array}$$

As a precondition for **P2** we only know that the message has not been deleted, but no information about the state of the mailbox is available, i.e., we do not know whether the mailbox is open or closed. Thus, the plan **P2** must contain 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 message x can be read and deleted. In contrast to the goal specification in S_{P1}, the specification of goals in S_{P2} specifies no temporary goals, but a *conjunctive* goal.

The third specification S_{P3} specifies an *iterative* plan reading all messages from sender *Joe* in the mailbox *mbox*. The specification of its preconditions and goals contains universally quantified formulae:

$$\mathbf{S_{P3}}: \quad \begin{array}{l} \mathsf{Plan}_{P3} \wedge open\_flag(mbox) = T \wedge \\ \forall x\,[\,sender(msg(x, mbox)) = Joe \\ \qquad \rightarrow delete\_flag(msg(x, mbox)) = F\,] \\ \rightarrow \Diamond\,\forall x\,[\,sender(msg(x, mbox)) = Joe \\ \qquad \rightarrow read\_flag(msg(x, mbox)) = T \wedge \\ \qquad\quad delete\_flag(msg(x, mbox)) = T\,] \end{array}$$

A restricted syntactic class of LLP formulae is used for the representation of plan specifications. For example, implicit negation of atomic formulae

occurs in implications. Furthermore, atomic formulae are equations assigning values to local variables appearing in terms of a certain syntactic structure. The term $msg(x, mbox)$ denotes a message in a mailbox $mbox$ at position x. Unary functions like $read\_flag$ and $delete\_flag$ represent features of a mailbox with respect to a particular message. The effects of actions are reflected in changed features.

5 Plan Determination: Efficient Retrieval of Plan Candidates

Assume as an example that the plan **P2** solving S_{P2} has to be generated from second principles, i.e., by reusing the plans **P1** or **P3** that are stored in the plan library.

The plan **P1** is a simple sequence containing the actions *type* and *delete*. The plan **P2** which we want to generate is a conditional plan, while **P3** is an iterative plan:

P1: $\boxed{ex(type(m, mybox)); ex(delete(m, mybox))}$

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

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

Plan determination has to answer the question of whether **P1** or **P3** are appropriate candidates to guide the planning process for **P2** and which of the two candidates should be preferred. A closer look at the plans reveals that they have a sequential subplan in common, cf. the framed formula. Apart from this, the plans differ mainly in the control structures they contain. Therefore, we are faced with two problems:

1. Existing frameworks from case-based reasoning fail in comparing such complex structures. There is no technique available in the literature on which the comparison and retrieval process of complex plans containing control structures can be based.

2. Even human experts are unable to compare such abstract logical descriptions of plans. It is by no means obvious whether we should take
   - plan **P1** and add a case analysis or
   - plan **P3** and remove the superfluous iterative control structure

   in order to obtain **P2**.

Both problems challenge a formal approach to the determination of reuse candidates from a plan library. The identification of **P1** and **P3** as appropriate reusable plans requires *abstraction* from

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

The basic effects of actions which cause a mailbox's features to be changed 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 description logic.

5.1 Description Logics as Query Languages

We define the *encoding scheme* ω as a mapping from the planning logic LLP to a description logic. The advantages in using a description logic as representation language for indices are obvious:

Description logics support a *structured* representation of *abstract* knowledge. As fragments of predicate logic they possess formal semantics [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.

Thus, indices are provided with clearly defined semantics. The monotonicity property of the encoding scheme ω can be proved, which ensures that existing solutions are found in the plan library when a *complete* retrieval algorithm is used.

The mathematical properties of various description logics are well understood. In particular, concept languages with decidable subsumption relations have been identified. Remember that retrieval from plan libraries must be efficient, i.e., the complexity of the retrieval algorithm must be investigated. The use of description logics possessing polynomial subsumption algorithms

ensures that the retrieval algorithm runs in polynomial time on the size of the plan library.

Usually, 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 description logics, case libraries are indexed on a more general *lattice structure* provided by the subsumption hierarchy [Koehler, 1994a].

Summarizing, it is possible to define retrieval algorithms with the following formal properties:

- **Correctness**: The retrieved plan entry meets the search criterion.
- **Completeness**: Existing solutions are in the retrieval set.
- **Complexity**: The retrieval algorithm runs in polynomial time.

The description logic $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991] is chosen as the target formalism of ω 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. There are two possibilities to obtain polynomial complexity: either to give up completeness or to restrict the description 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 a plan library, the incompleteness of the retrieval algorithm leads to the following problems:

- Existing solutions may be not found in the library. This can lead to an undesirable computational overhead in second-principles planning because the system does not reuse the best available plan during problem solving.
- Uncontrolled growth of the plan library may occur. Plan specifications with the same indices are added to the library because the incomplete subsumption algorithm is unable to recognize the equivalence of the concepts representing the indices.

Therefore, we decided to restrict concept descriptions to a *normal form* for which a sound, complete and polynomial subsumption algorithm exists. We define so-called *admissible concepts* as a subset of $\mathcal{ALC}$ that are consistent

concept descriptions in conjunctive normal form. They are built only 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. This simplifies the computation of subsumption relationships between concepts. To determine subsumption in a terminology, the relation between the extensions of concepts is symbolically evaluated. First, the relevant part of a terminology has to be transformed in a normal form. Secondly, constraints must be propagated and inconsistencies must be recognized. Finally, the resulting expressions are structurally compared, cf. [Nebel, 1990]. The restriction of ω's target formalism to admissible concepts makes the first and second step superfluous and reduces the computation of subsumption to a structural comparison of the concepts. A normal form is already given and no inconsistencies or constraints between concepts may occur. In fact, admissible concepts define a subset of propositional logic. Primitive components can be treated as atomic units during the computation of subsumption because there is no need to expand them further. Thus, 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*[3]*:*

$$\begin{aligned} z \sqsubseteq x, z \sqsubseteq y &\rightarrow z \sqsubseteq x \wedge y \\ x \sqsubseteq z &\rightarrow x \wedge y \sqsubseteq z \\ x \sqsubseteq z, y \sqsubseteq z &\rightarrow x \vee y \sqsubseteq z \\ z \sqsubseteq x &\rightarrow z \sqsubseteq x \vee y \\ x &\sqsubseteq x \end{aligned}$$

Theorem 2 *SUBS is sound and complete, and decides the subsumption relation in polynomial time for admissible concepts.*

The proof is straightforward and can be found in [Koehler, 1994e].

5.2 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.

LLP plan specifications are a restricted class of temporal logic formulae containing the modal operator $\Diamond$. In order to map them to concept descriptions

[3]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 $\mathcal{ALC}$.

they are equivalently translated into first-order predicate logic using a relational translation method for modal logics as for example introduced in [Ohlbach, 1991]. A function π is defined that translates an LLP formula into a formula in CPL, a constraint predicate logic. Below, a subset of π is shown, which is needed for the translation of plan-specification formulae:

$$\begin{aligned}
\pi[x, w] &:= x \quad \text{for } x \text{ a global variable} \\
\pi[x, w] &:= x(w) \quad \text{for } x \text{ a local variable} \\
\pi[f(t_1, \ldots, t_n), w] &:= f(\pi[t_1, w], \ldots, \pi[t_n, w]) \\
\pi[P(t_1, \ldots, t_n), w] &:= P(\pi[t_1, w], \ldots, \pi[t_n, w]) \\
\pi[F \wedge G, w] &:= \pi[F, w] \wedge \pi[G, w] \\
\pi[\Diamond F, w] &:= \exists v \big(w \geq v \wedge \pi[F, v] \big)
\end{aligned}$$

Using the method developed in [Frisch and Scherl, 1991], which has been extended to LLP in [Koehler and Treinen, 1993], the formulae resulting from the translation of modal operators can be considered as constraints. As an example, let us consider the encoding of the goal specification of S_{P1}

$$\begin{aligned}
\Diamond [\, & read_flag(msg(m, mybox)) = T \wedge \\
& \Diamond [\, delete_flag(msg(m, mybox)) = T \,] \,]
\end{aligned}$$

An application of the translation rules of π to this LLP formula leads to a formula in the logic CPL. The function π translates local variables into unary functions mapping an interval to the value of the local variable in this interval. Modal operators are translated into constraints reflecting the accessibility relations defined over intervals. Observe that function and predicate symbols are so-called rigid designators, i.e., their interpretation is fixed and thus, they do not have to be equipped with an interval argument during the translation. We obtain the following CPL formula:

$$\begin{aligned}
\exists w_1, w_2 \, [\, & w_0 \geq w_1 \wedge w_1 \geq w_2 \wedge \\
& read_flag(msg(m, mybox(w_1))) = T \wedge \\
& delete_flag(msg(m, mybox(w_2))) = T \,]
\end{aligned}$$

A constraint formula $\mathcal{C}$ can be separated from the constraint-free part of the formula according to the following rule [Frisch and Scherl, 1991]:

$$[\exists y \, \mathcal{C} \wedge \phi] \wedge \psi \equiv \exists y \, \mathcal{C} \wedge [\phi \wedge \psi]$$

In a next step, information about the ordering of temporary subgoal states is eliminated from the formula, which implements the process of temporal

abstraction. Obviously, this transformation leads to a weaker logical formula. In order to ensure the correctness of this transformation, the monotonicity property (Condition 1) has to be proved for it, i.e., the elimination of a set of constraints resulting from the translation of $\Diamond$ operators has to preserve an existing subset relationship between the models of the formulae.[4] The elimination of the constraint formulae from the example formula leads to

$$read\_flag(msg(m, mybox(w_1))) = T \wedge delete\_flag(msg(m, mybox(w_2))) = T$$

Now, each atomic formula is mapped to a primitive component, i.e., an existential role restriction of the form $\exists R.C$, while the logical conjunction $\wedge$ is mapped to the concept intersection $\sqcap$.

Let us have a closer look at the syntactical structure of the atomic formula $read\_flag(msg(m, mybox(w_1))) = T$. First, the constant T is replaced by an existentially quantified variable y using the rule $P(a) \rightarrow \exists y\ P(y)$. This implements an abstraction from specific objects occurring in the specification formulae. Furthermore, we add a unary predicate $true(y)$ expressing the sort information for the variable y. We treat the equality predicate like an ordinary predicate P and thus obtain

$$\exists y\ P(read\_flag(msg(m, mybox(w_1))), y) \wedge true(y)$$

Implicit or explicit universal quantification is replaced by existential quantification according to the rule $\forall x\ P(x) \rightarrow \exists x\ P(x)$, which implements, e.g., a process of abstraction from universally quantified goals.

With that we have obtained a formula of the form $\phi_C(x) : \exists\, x\ \exists\, y\, P(x, y) \wedge Q(y)$ to which a concept $C : \exists P.Q$ corresponds. A model of the formula $\phi_C(x)$ is a model of the concept C and vice versa. In particular, C is unsatisfiable if and only if $\phi_C(x)$ is unsatisfiable [Hollunder *et al.*, 1990].

The structure of the term $read\_flag(msg(m, mybox(w_1)))$ is reflected in the composition of roles. The unary function *mybox* is of type $interval \rightarrow mailbox$ and is abstracted by a binary relation $interval \times mailbox$. The binary function *msg* is of type $mailbox \times integer \rightarrow message$, i.e., it takes a mailbox and an integer as arguments and returns the message that can be found in the mailbox at the position indicated by the integer. Thus, this function is abstracted by the composition of binary relations $mailbox \times integer \circ integer \times message$. The unary function *read_flag* is of type $message \rightarrow boolean$,

[4]The proof can be found in [Koehler, 1994e]. It is not presented here because it does not directly contribute to the aim of this paper which is to present a logic-based framework for planning from second principles.

i.e., we abstract it by a binary relation of type $message \times boolean$. Consequently, for the whole term the composition of binary relations

$$\underbrace{interval \times mailbox}_{\text{mailbox}} \circ \underbrace{mailbox \times integer}_{\text{position}} \circ \underbrace{integer \times message}_{\text{message}} \circ \underbrace{message \times boolean}_{\text{read\_flag}}$$

is obtained, leading to the following role chain

$$\exists \, \text{mailbox} \circ \text{position} \circ \text{message} \circ \text{read\_flag}.\text{TRUE}$$

the value of which is existentially restricted to the concept TRUE.

After the encoding process has been completed, the conjunctive normal form of the indices 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. Besides this, plan specifications are often given in a conjunctive normal form[5] and therefore, this costly operation is not necessary in most cases.

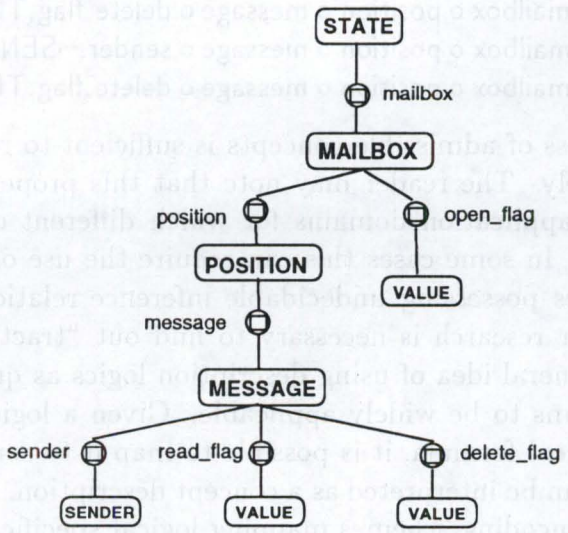

Figure 3: Subset of the mail domain terminology

[5]Existing second-principles planning systems are limited to deal with state descriptions restricted to conjunctions of literals. This means that the PHI planner poses additional requirements on the encoding scheme unknown in the usual STRIPS-based planners.

Figure 3 shows a subset of the mail-domain terminology that is used for the encoding of the example specifications. Nested terms, which are used to represent features of mailboxes in the source formalism LLL, map to role chains in the target formalism.[6]

The encoding scheme ω used in MRL leads to the following encodings of the specifications S_{P1} to S_{P3}.

$\omega(pre_{P1})$: $\exists$ mailbox $\circ$ open_flag.TRUE $\sqcap$
$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.FALSE

$\omega(goal_{P1})$: $\exists$ mailbox $\circ$ position $\circ$ message $\circ$ read_flag.TRUE $\sqcap$
$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.TRUE

$\omega(pre_{P2})$: $\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.FALSE

$\omega(goal_{P2})$: $\exists$ mailbox $\circ$ position $\circ$ message $\circ$ read_flag.TRUE $\sqcap$
$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.TRUE

$\omega(pre_{P3})$: $\exists$ mailbox $\circ$ open_flag.TRUE $\sqcap$
[$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ sender.$\neg$SENDER $\sqcup$
$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.FALSE]

$\omega(goal_{P3})$: [$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ sender.$\neg$SENDER $\sqcup$
$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.TRUE] $\sqcap$
[$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ sender.$\neg$SENDER $\sqcup$
$\exists$ mailbox $\circ$ position $\circ$ message $\circ$ delete_flag.TRUE]

The expressiveness of admissible concepts is sufficient to represent the mail domain adequately. The reader may note that this property may not generalize to other application domains for which different encoding schemes must be defined. In some cases this can require the use of more expressive concept languages possessing undecidable inference relations as target formalisms. Further research is necessary to find out "tractable" application domains. The general idea of using description logics as query languages to case libraries seems to be widely applicable. Given a logical description of a case, i.e., a logical formula, it is possible to map it to some weaker logical formula, which can be interpreted as a concept description. Nevertheless, the development of encoding schemes mapping logical specifications to concept descriptions is a creative process. It's mechanization is an interesting subject for further research.

The encoding scheme used in MRL satisfies the monotonicity property as stated in Theorem 3. Thus, the retrieval algorithm is guaranteed to find

[6] In principle, it seems to be possible to automatically generate the encoding terminology from the signature used in the source formalism.

existing solutions when a complete subsumption algorithm is used.

Theorem 3 *If* $pre_{new} \rightarrow pre_{old}$ *and* $goal_{old} \rightarrow goal_{new}$ *then* $\omega(pre_{new}) \sqsubseteq \omega(pre_{old})$ *and* $\omega(goal_{old}) \sqsubseteq \omega(goal_{new})$.

The proof can be found in [Koehler, 1994e]. It relies on the model-theoretic semantics the logics LLP, CPL and $\mathcal{ALC}$ possess.

5.3 Weak and Strong Classification

The results of the encoding process are the admissible concepts $\omega(pre)$ and $\omega(goal)$ from which the index of a plan is obtained as the pair $\langle \omega(pre), \omega(goal) \rangle$. Indices are considered as new representational primitives in the description logic. The computation of the subsumption relation between indices is reduced to computing the subsumption relation between the concepts encoding goals and preconditions as defined in Definition 3:

Definition 3 $\langle \omega(pre_{old}), \omega(goal_{old}) \rangle$ *is subsumed by* $\langle \omega(pre_{new}), \omega(goal_{new}) \rangle$ *if and only if*

$$\omega(pre_{new}) \sqsubseteq \omega(pre_{old}) \quad \textit{and} \quad \omega(goal_{old}) \sqsubseteq \omega(goal_{new}).$$

Now, the *retrieval* of a plan from the plan library is formalized as follows: Given a new plan specification, its index is computed first. Then, this index is classified in the plan library. Two classification operations are available:

- **Strong classification** which classifies the new index by computing the subsumption relation between encodings of preconditions and goals

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

  The result of strong classification determines the position of the new index in the plan library according to the subsumption of indices as defined in Definition 3. 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 achieve all of the current goals.

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

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

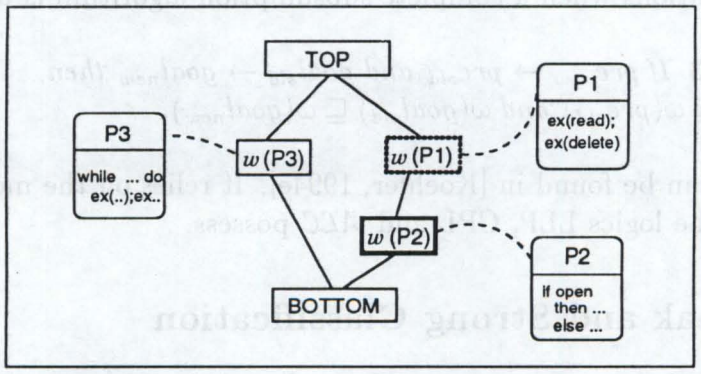

Figure 4: The example library

Figure 4 shows the small example library obtained for the three plan specifications under consideration: Obviously, the index $\omega(P2)$ of S_{P2} subsumes only the bottom concept, i.e., strong classification fails in retrieving a candidate plan. Therefore, weak classification is activated searching for an $\omega(pre_{old})$ that subsumes $\omega(pre_{P2})$ or for an $\omega(goal_{old})$ that is subsumed by $\omega(goal_{P2})$. Weak classification of the preconditions fails as well, while weak classification of the goals is successful for $\omega(goal_{P1})$ since $\omega(goal_{P1}) \sqsubseteq \omega(goal_{P2})$ holds.[7] Therefore, the plan stored in the plan entry belonging to $\omega(S_{P1})$ is activated as a reuse candidate. Since strong classification failed we know that this plan cannot represent a solution to the current planning problem S_{P2}. We expect it to achieve all of the current goals, but we know that its preconditions are not satisfied in the current initial state. Thus, plan refitting has to start as will be described in Section 6.

5.4 Ranking of Plan Entries

Only one candidate plan has been retrieved from the plan library in the example under consideration. But in general, strong as well as weak classification can retrieve several appropriate reuse candidates. Consequently, a *ranking* is needed for the candidates in order to determine the best.

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

[7] In a working system it seems to be a good restriction to implement only one of the possible approaches to weak classification in order to improve retrieval efficiency. Here, we discuss both possibilities in order to present how retrieval based on classification works. MRL applies weak classification only to preconditions, i.e., it requires plans to be applicable in the current initial state as a heuristic to reduce the refitting effort during plan modification.

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 can be eliminated from the reused plan by making attempts at optimizing it. Thus, the ranking of the candidates is based on an estimation of the *optimization effort* for each candidate, i.e., the number of superfluous actions that have to be eliminated from the candidate plan. 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 4 *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 occur in a concept c is denoted by $\mathsf{P}[c]$. The cardinality of the set $\mathsf{P}[c]$ is as usually denoted by $\mid \mathsf{P}[c] \mid$.*
The optimization effort for each candidate is defined as

$$OPT_{\omega(goal_{old_i})} = \mid \mathsf{P}[\omega(goal_{old_i})] \setminus \mathsf{P}[\omega(goal_{new})] \mid$$

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

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

Weak classification selects plans from the plan library that are either supposed to be applicable in the initial state or to achieve the desired goals, i.e., we have to expect that every candidate has to be modified. Consequently, the heuristic estimates the *modification effort* for each candidate in the retrieval set. It 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 retrieval set and 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 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.

Definition 5 *The estimated success rate for each candidate is defined as:*

$$MOD_{\omega(goal_{old_i})} = \mid \mathsf{P}[\omega(goal_{old_i})] \cap \mathsf{P}[\omega(goal_{new})] \mid$$

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 estimated as too expensive. In this situation, plan determination reports a failure and planning from scratch with the PHI planner is activated.

The ranking heuristics guide the interaction between planning from first and planning from second principles, cf. Figure 5.

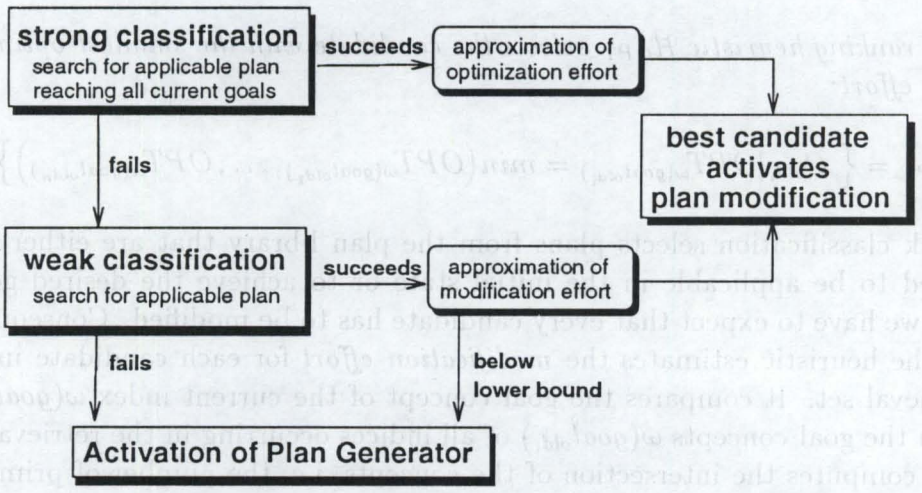

Figure 5: Interaction between first and second-principles planning

6 Plan Modification: Provably Correct Plans

Plan modification is based on deductive inference processes which lead to modified plans that are provably correct. As introduced in Section 3 it proceeds in two phases. First, *plan interpretation* computes a plan skeleton and second, *plan refitting* completes the plan skeleton to a correct plan satisfying the current specification.

In the following, we apply the formal approach to plan modification as defined in Section 3 to the example under consideration and discuss deductive plan modification in MRL.

6.1 Plan Interpretation

Plan interpretation receives two sources of input:

1. the current plan specification for which a plan has to be generated
2. the plan entry containing the best reusable plan which the determination phase could identify in the plan library.

It takes the plan specification from the plan entry and the current plan specification and attempts to prove the required relations between preconditions and goals:

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

During the proof, knowledge concerning regularities in the planning domain is applied that can be extracted, e.g., from the action axiom schemata available to the planner. For example, from the axiomatization of the *save*-command

$$\begin{aligned} \forall x\,[\,&open\_flag(mailbox) = T \wedge delete\_flag(msg(x, mailbox)) = F \wedge \\ &ex(save(x, file, mailbox)) \\ \rightarrow \circ\,[\,&file(msg(x, mailbox)) = file \wedge save\_flag(msg(x, mailbox)) = T\,]\,] \end{aligned}$$

we can derive the following consequences that can be used as additional non-logical axioms

$$save\_flag(msg(x, mailbox)) = T \rightarrow file(msg(x, mailbox)) = file$$

$$file(msg(x, mailbox)) = file \rightarrow save\_flag(msg(x, mailbox)) = T$$

reflecting the relationship between the atomic effects of this command. Whenever the *save_flag* of a message has been set to T then there must be a file in

which the message has been saved and vice versa. This makes plan interpretation more flexible than syntactic matching because it can identify plans as reusable, even if their specifications are syntactically different.

Plan interpretation attempts to prove that $S_{old} \rightarrow S_{new}$ holds in the domain theory using the LLP sequent calculus [Biundo and Dengler, 1994].[8] In the example, it builds the sequent $S_{P1} \Rightarrow S_{P2}$ and applies the derived rule *rule_one*, which extracts the starting sequents for the subproofs of preconditions and goals:

$$\frac{\mathsf{Plan}_{\mathsf{new}} \Rightarrow \mathsf{Plan}_{\mathsf{old}} \qquad pre_{new} \Rightarrow pre_{old} \qquad goal_{old} \Rightarrow goal_{new}}{\mathsf{Plan}_{\mathsf{old}} \wedge pre_{old} \rightarrow goal_{old} \Rightarrow \mathsf{Plan}_{\mathsf{new}} \wedge pre_{new} \rightarrow goal_{new}}\, rule\_one$$

In the example, the proof of the relations between the *preconditions* requires Sequent 1 to be proved:

$$\begin{array}{l} delete\_flag(msg(x, mbox)) = F \\ \Rightarrow open\_flag(mybox) = T \wedge delete\_flag(msg(m, mybox)) = F \end{array} \quad (1)$$

Starting point for the *goal* proof is Sequent 2:

$$\begin{array}{l} \Diamond\,[\,read\_flag(msg(m, mybox)) = T \wedge \\ \qquad \Diamond[delete\_flag(msg(m, mybox)) = T\,]\,] \\ \Rightarrow \Diamond\,[\,read\_flag(msg(x, mbox)) = T \wedge \\ \qquad delete\_flag(msg(x, mbox)) = T\,] \end{array} \quad (2)$$

Special-purpose *proof tactics* guide the proof attempt in the LLP sequent calculus. They run in polynomial time on the length of the input formula. On the one hand, this enables plan interpretation to compute an *entry point* into the search space of plans *efficiently.* On the other hand, this implies that the tactic is *incomplete* in the sense that it cannot compute a *maximal plan skeleton* which has been shown to be a PSPACE-hard problem in [Nebel and Koehler, 1993a]. Figure 6 partially sketches the tactic *precond_tac* that is used in MRL.[9] The tactic specifies a well defined ordering of deduction rule applications. It is composed of *tacticals* [Biundo and Dengler, 1994] like *iterate_rule*, *apply_rule_strict*, and *call_tac*. Each tactical specifies a specific mode of rule or tactic applications.

[8] A general introduction into sequent calculi can be found in [Gallier, 1987; Wallen, 1989].

[9] Subtactics dealing with universally quantified formulae are discussed in [Koehler, 1994c].

```
precond_tac(pre_seq, List_of_Axioms):-
      or_else([call_tac(separate_new,[pre_seq], List_of_Axioms),
                  call_tac(expand_final_new,[pre_seq], List_of_Axioms)]).
separate_new(pre_seq, List_of_Axioms):-
      apply_rule_strict(l∨,[pre_seq],[new1,rest_new]),
      iterate_rule(l∧,[new1],[new1_atom]),
      call_tac(expand_subtree,[new1_atom],List_of_Axioms),
      call_tac(precond_tac,[rest_new]).
expand_final_new(pre_seq, List_of_Axioms):-
      iterate_rule(l∧,[pre_seq],[new1_atom]),
      call_tac(expand_subtree,[new1_atom],List_of_Axioms).
expand_subtree(new1_atom, List_of_Axioms):-
      or_else([call_tac(split_further_subtree,[new1_atom],List_of_Axioms),
                  call_tac(close_final_subtree,[new1_atom],List_of_Axioms)]).
split_further_subtree(new1_atom, List_of_Axioms):-
      apply_rule_strict(split_proof,[new1_atom],[old1,remaining_old]),
      iterate_rule(r∧,[old1],List_of_Subtree_Leaves),
      call_tac(close_leaves,List_of_Subtree_Leaves,List_of_Axioms),
      call_tac(expand_subtree,[remaining_old],List_of_Axioms).
close_final_subtree(new1_atom, List_of_Axioms):-
      iterate_rule(r∧,[new1_atom],List_of_Subtree_Leaves),
      call_tac(close_leaves,List_of_Subtree_Leaves,List_of_Axioms).
```

Figure 6: Sketch of the tactic for the precondition proof

The tactical *apply_rule_strict* applies the rule specified in its first argument to a sequent specified in its second argument and returns as a result the sequent specified in its third argument. It fails when the specified rule is not applicable to the input sequent. The tactical *apply_rule* works like *apply_rule_strict* with the difference that it always succeeds, i.e., if the specified rule is not applicable to the input sequent this sequent is returned unchanged. The tactical *iterate_rule* repeats a rule application as long as possible, while the tactical *call_tac* calls another tactic.

The tactic *precond_tac* is able to deal with disjunctive and conjunctive preconditions and applies the following sequent rules:

- $\dfrac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta}\, l\wedge$
- $\dfrac{\Gamma \Rightarrow A, \Delta \qquad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta}\, r\wedge$
- $\dfrac{\Gamma, A \Rightarrow \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow \Delta}\, l\vee$
- $\dfrac{\Gamma \Rightarrow A, B, \Delta}{\Gamma \Rightarrow A \vee B, \Delta}\, r\vee$

- $\dfrac{\Gamma \Rightarrow A \qquad \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow A \vee \Delta}$ *split_proof*

The precondition proof for the example sequent is very simple because no disjunctive preconditions occur in the sequent expressing uncertainty about the initial state. The first rule that is successfully applied to Sequent 1 is rule $r\wedge$. It leads to Sequents 3 and 4:

$$delete\_flag(msg(x, mbox)) = F \Rightarrow open\_flag(mybox) = T \quad (3)$$

$$delete\_flag(msg(x, mbox)) = F \Rightarrow delete\_flag(msg(m, mybox)) = F \quad (4)$$

Sequent 4 can be closed, i.e., it leads to Axiom A1 using the substitution $\{x \rightarrow m, mbox \rightarrow mybox\}$.

(A1) $\boxed{delete\_flag(msg(x, mbox)) = F \Rightarrow delete\_flag(msg(x, mbox)) = F}$

In order to obtain an appropriate instantiation of the reused plan, variables in the reused specification S_{old} are substituted by terms which occur in the current specification S_{new}. Furthermore, different variables must be mapped to different terms, i.e., the substitutions must be *injective*. Injectivity may not always be required but it is a save condition ensuring that a proper instance of the reuse candidate is computed during the proof. The reader may note that an instantiation of sequents during a sequent proof is only possible when quantifier rules are applied. Plan specification formulae are implicitly universally quantified, i.e., when proving $S_{old} \Rightarrow S_{new}$ in the sequent calculus we remove the universal quantifiers using the rules $l\forall$ and $r\forall$ and have to "guess" the appropriate instantiation. Of course, this is unacceptable in an implemented prover due to the resulting computational overhead. Therefore, the instantiation is delayed until we know which instantiation is appropriate, i.e., which one will lead to a proof of the sequent. The restrictions we pose on the instantiations of the leaf sequents ensure that only those instantiations are computed that can be introduced with the help of the quantifier rules $l\forall$ and $r\forall$:

- $\dfrac{\Gamma, A[c/x] \Rightarrow \Delta}{\Gamma, \forall x A \Rightarrow \Delta}\ l\forall$

- $\dfrac{\Gamma \Rightarrow \Delta, A[a/x]}{\Gamma \Rightarrow \Delta, \forall x A}\ r\forall$

Eigenvariable condition: a must not occur in the conclusion of $r\forall$

The tactic for the *goal proof* is shown in Figure 7. It has to cope with modal operators and therefore additionally uses the following sequent rules:

```
goal_tac(goal_seq,List_of_Axioms):-
        apply_rule_strict(l◇,[goal_seq],[first_old]),
        apply_rule_strict(r◇,[first_old],[first_old_and_new]),
        iterate_rule(l∧,[first_old_and_new],[first_atom_left]),
        iterate_rule(r∧,[first_atom_left],[first_atom_right,remaining_seq]),
        call_tac(close_leaves,[first_atom_right],List_of_Axioms),
        call_tac(expand_goal_subtree,[remaining_seq],List_of_Axioms),
        call_tac(goal_tac,[remaining_seq],List_of_Axioms).
expand_goal_subtree(remaining_seq,List_of_Axioms):-
        apply_rule_strict(r◇,[remaining_seq],[next_new]),
        iterate_rule(r∧,[next_new],[next_new_atom]),
        call_tac(close_leaves,[next_new_atom],List_of_Axioms),
        call_tac(expand_goal_subtree,[next_new],List_of_Axioms).
```

Figure 7: Sketch of the tactic for the goal proof

$$\bullet \ \frac{\Gamma^*, A \Rightarrow \Delta^*}{\Gamma, \Diamond A \Rightarrow \Delta} \, l\Diamond \qquad \bullet \ \frac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow \Diamond A, \Delta} \, r\Diamond$$

With Γ^* and Δ^*: $\Gamma^* \stackrel{\text{df}}{=} \{\Box B | \Box B \in \Gamma\}$ and $\Delta^* \stackrel{\text{df}}{=} \{\Diamond B | \Diamond B \in \Delta\}$.

The tactic proceeds recursively over the *sometimes* operators in both goal specifications in order to compare every temporary subgoal state specified in $goal_{old}$ with each of the temporary subgoal states from $goal_{new}$.

The proof of the goal sequent proceeds straightforwardly with the help of the tactic.[10] The tactic applies rule $l\Diamond$ to Sequent 2 followed by rule $r\Diamond$ and obtains Sequent 5:

$$\begin{array}{l} read\_flag(msg(m, mybox)) = T \wedge \\ \quad \Diamond\,[\,delete\_flag(msg(m, mybox)) = T\,] \\ \Rightarrow \\ read\_flag(msg(x, mbox)) = T \wedge delete\_flag(msg(x, mbox)) = T \end{array} \tag{5}$$

Now, the tactic applies rule $l\wedge$ followed by rule $r\wedge$ to Sequent 5 which leads to Sequents 6 and 7

$$\begin{array}{l} read\_flag(msg(m, mybox)) = T, \\ \quad \Diamond\,[\,delete\_flag(msg(m, mybox)) = T\,] \\ \Rightarrow read\_flag(msg(x, mbox)) = T \end{array} \tag{6}$$

[10] A more complex example dealing with universally quantified subgoals can be found in [Koehler, 1994c].

$$
\begin{aligned}
&read\_flag(msg(m, mybox)) = T, \\
&\quad \Diamond\,[\,delete\_flag(msg(m, mybox)) = T\,] \\
&\Rightarrow delete\_flag(msg(x, mbox)) = T
\end{aligned}
\tag{7}
$$

Sequent 6 can also be closed under the substitution $\{x \rightarrow m, mbox \rightarrow mybox\}$, i.e., the current subgoal $read\_flag(msg(x, mbox)) = T$ has been successfully proved by the tactic:

(A2) $\boxed{read\_flag(msg(x, mbox)) = T \Rightarrow read\_flag(msg(x, mbox)) = T}$

The tactic proceeds on Sequent 7 and removes the remaining $\Diamond$ operator with the help of rule $l\Diamond$ which leads to Sequent 8

$$
delete\_flag(msg(m, mybox)) = T \Rightarrow \emptyset \tag{8}
$$

The formula $delete\_flag(msg(x, mbox)) = T$ from the succedent of Sequent 7 disappears in Sequent 8 because it does not occur in the scope of a $\Diamond$ operator. Thus, the tactic fails in proving the remaining subgoal. The reason for this failure is obvious: The current goal specification requires the two subgoals to be achieved in the same state, while the reused goal specification only requires the two subgoals to be achieved one after the other. Of course, deleting a mail preserves the effect that the mail has been read, i.e., the reused plan that first reads the mail and then deletes it also leads to a final state where the mail has been read and deleted. But we have no way to derive this fact from the original plan specification formula. This is a motivation for a completion process of plan specification formulae that is described in Section 7.

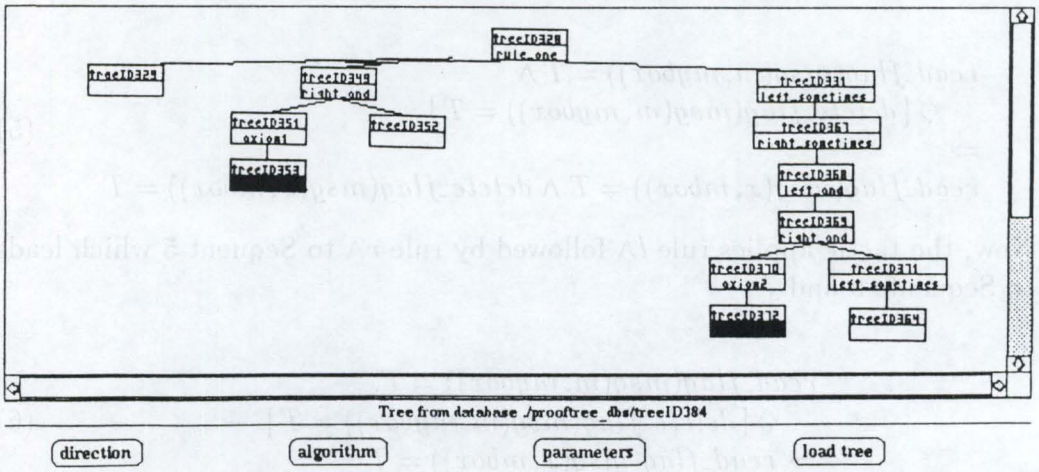

Figure 8: Visualization of the plan interpretation phase in MRL

Figure 8 illustrates the deduction tree which is constructed by the tactics during the example proof. The black nodes designate two axioms that were found. Axiom *axiom1* represents one atomic precondition of the reused plan that holds in the current initial state, while *axiom2* represents one atomic current goal that is achieved by the reused plan.
The white nodes (treeID352 and treeID364) visualize leaves that could not be closed during the proof attempt, i.e., the tactics failed in proving that the plan is applicable in the current initial state and that it achieves all of the currently required goals. Thus, plan refitting must begin.

6.2 Plan Refitting

The proof tactics are designed always to terminate. In addition, they are considered as decision procedures: If a tactic does not result in a proof tree, it is assumed that no proof is possible and that a falsifying valuation for some of the leaves has been obtained. Two situations are possible after the termination of the proof tactics in the sequent calculus:

1. A proof tree has been constructed, i.e., the leaves of the tree describe a set of logical axioms from which the formula follows. In this case the formula was proved to be valid.

2. No proof tree has been found and the assumption is made that no proof is possible and that a counter-example tree has been constructed.

This assumption is a save condition ensuring the soundness of plan modification. Remember that the tactics are incomplete, i.e., when a tactic terminates with a failure it might either be the case that the formula is invalid or that the formula is valid, but the tactics failed to find a proof.

Assuming that the formula is invalid ensures that the correctness of a plan is verified during plan refitting. Thus it prevents a reuse of plans that are not provably correct with respect to the current plan specification.

The proof tactics guarantee that the leaves of a counter-example tree contain only atomic formulae. The falsifying valuation makes

- all old atomic goals (in the example from S_{P1}) true, however some of the atomic formulae which describe current goals (in the example from S_{P2}) are valued as false. These falsified goals are interpreted as those current goals that are not achieved by the reused plan (in the example by **P1**).

- all atomic formulae describing current preconditions (in the example from S_{P2}) true, but some of the old preconditions (in the example

from S_{P1}) false. These falsified preconditions are interpreted as those preconditions of the reused plan (in the example of **P1**) that do not hold in the current initial state.

Therefore, plan **P1** must be modified by constructing a plan skeleton from it. First, the plan is instantiated with the substitutions computed during plan interpretation leading to

P1': $ex(type(x, mbox)); ex(delete(x, mbox))$

Plan refitting concludes from the two non-axiom leaves that the (instantiated) precondition $open\_flag(msg(mbox)) = F$ required by **P1'** does not hold in the initial state and that the current goal $delete\_flag(msg(x, mbox)) = T$ is not achieved by it. Furthermore, **P1'** achieves a subgoal

$$delete\_flag(msg(m, mybox)) = T$$

that is not contained in one of the axioms constructed during the goal proof. Thus, plan refitting concludes that the action $ex(delete(m, mybox))$ achieving this subgoal[11] is (at least at the current position where it occurs) superfluous and can be removed from the plan skeleton. Table 9 summarizes the analysis of plan interpretation that is performed by plan refitting for the example under consideration. The necessary modification operations are derived based on these results.

preconditions		
old	$delete\_flag(msg(m, mybox)) = F$	
new	$delete\_flag(msg(x, mbox)) = F$	Axiom
old	$open\_flag(msg(mybox)) = T$	no
new	–	Axiom

goals		
old	$read\_flag(msg(m, mybox)) = T$	
new	$read\_flag(msg(x, mbox)) = T$	Axiom
old	–	no
new	$delete\_flag(msg(x, mbox)) = T$	Axiom
old	$delete\_flag(msg(m, mybox)) = T$	no
new	–	Axiom

Figure 9: Analysis of the plan interpretation phase

The following modification operations have to be performed on the instantiated plan **P1'**:

[11] Knowledge about relations between actions and effects is stored in the plan entries, cf. Section 7.

1. A planvariable has to be introduced in front of the reused plan. It represents a subplan achieving the missing *precondition.*

2. The *superfluous* action is removed from the plan skeleton.

3. A planvariable representing the subplan for the open *subgoal* must be introduced into the plan skeleton. In order to determine the position in the skeleton where this planvariable has to be added, the current goal state specification must be analyzed with the help of the PHI planner.

The current plan specification is instantiated with the preliminary plan skeleton **P1″**. It serves as a starting point for the planner:

P1″: $\mathsf{Plan}_1 ; ex(type(x, mbox))$

$$\mathbf{S_{P2'}}: \quad \begin{array}{l} \mathsf{Plan}_1 ; ex(type(x, mbox)) \wedge delete\_flag(msg(x, mbox)) = F \\ \rightarrow \Diamond [\, read\_flag(msg(x, mbox)) = T \wedge \\ \qquad delete\_flag(msg(x, mbox)) = T \,] \end{array}$$

In a first step, a subplan to replace Plan_1 has to be generated. Plan refitting applies the rule *effect_intro* [Biundo and Dengler, 1994] and introduces the missing precondition as the new subgoal $goal_{new}$:

$$\frac{pre, \mathsf{Plan}_1 \Rightarrow \Diamond[goal_{new} \wedge \circ F \wedge pre'] \qquad pre', \mathsf{Plan}_2 \Rightarrow \Diamond goal}{pre, \mathsf{Plan}_1 ; \mathsf{Plan}_2 \Rightarrow \Diamond goal} \; effect\_intro$$

It obtains the two subplan specifications 9 and 10 where Plan_2 is instantiated with the action $ex(type(x, mbox))$ taken from the plan skeleton. The preconditions pre' are instantiated with $delete\_flag(msg(x, mbox)) = F$ by applying the mechanism for the computation of frame conditions that is provided by the PHI planner [Biundo *et al.*, 1992]:

$$\begin{array}{l} delete\_flag(msg(x, mbox)) = F, \mathsf{Plan}_1 \\ \Rightarrow \Diamond [\, open\_flag(msg(mbox)) = T \wedge \circ F \wedge \\ \qquad delete\_flag(msg(x, mbox)) = F \,] \end{array} \tag{9}$$

$$\begin{array}{l} open\_flag(msg(mbox)) = T \wedge delete\_flag(msg(x, mbox)) = F, \\ ex(type(x, mbox)) \\ \Rightarrow \Diamond [\, read\_flag(msg(x, mbox)) = T \wedge \\ \qquad delete\_flag(msg(x, mbox)) = T \,] \end{array} \tag{10}$$

The proof of the subplan specification 9 leads to a conditional plan because there is no action available in the domain axiomatization that achieves the required goal under the given precondition. Plan refitting applies the rule *if_intro* [Biundo and Dengler, 1994] to insert the case analysis:

$$\frac{pre, \mathbf{if}(cond, \mathsf{Plan_A}, \mathsf{Plan_B}) \Rightarrow \Diamond goal}{pre, \mathsf{Plan} \Rightarrow \Diamond goal}\ if\_intro$$

In the example, the planvariable $\mathsf{Plan_1}$ is instantiated with a case analysis over the state of the mailbox, which is the missing precondition that plan interpretation failed to prove:

$\mathsf{Plan_1} :=$ **if** $open\_flag(mbox) = T$ **then** $\mathsf{Plan_3}$
else $\mathsf{Plan_4}$

Applying the rule *if_splitting* [Biundo and Dengler, 1994] to Sequent 9, plan refitting obtains the following subplan specifications:

$$\frac{pre, cond, \mathsf{Plan_A} \Rightarrow \Diamond goal \qquad pre, \neg cond, \mathsf{Plan_B} \Rightarrow \Diamond goal}{pre, \mathbf{if}(cond, \mathsf{Plan_A}, \mathsf{Plan_B}) \Rightarrow \Diamond goal}\ if\_splitting$$

$$\begin{array}{l} delete\_flag(msg(x, mbox)) = F, \underline{open\_flag(mbox) = T}, \mathsf{Plan_3} \\ \Rightarrow \Diamond [\underline{open\_flag(mbox) = T} \wedge \circ F \wedge \\ \quad \underline{delete\_flag(msg(x, mbox)) = F}] \end{array} \tag{11}$$

$$\begin{array}{l} delete\_flag(msg(x, mbox)) = F, \neg open\_flag(mbox) = T, \mathsf{Plan_4} \\ \Rightarrow \Diamond [open\_flag(mbox) = T \wedge \circ F \wedge \\ \quad delete\_flag(msg(x, mbox)) = F] \end{array} \tag{12}$$

$\mathsf{Plan_3}$ is instantiated with the empty action $ex(empty\_action)$ because the desired subgoal already holds in the initial state (see the underlined formulae in Sequent 11), while $\mathsf{Plan_4}$ is instantiated with the action instance $ex(mail(mbox))$ which opens the mailbox and starts a mail session:

$$\begin{array}{l} \forall x [open\_flag(mailbox) = F \wedge ex(mail(mailbox)) \\ \quad \rightarrow \circ\, open\_flag(mailbox) = T]^{12} \end{array}$$

Thus, the following conditional plan is obtained as an instantiation of $\mathsf{Plan_1}$:

$\mathsf{Plan_1} :=$ **if** $open\_flag(mbox) = T$ **then** $ex(empty\_action)$
else $ex(mail(mbox))$

The proof of the subplan specification 10 proceeds as an interleaved process of plan generation and plan verification. First, a tactic for the ordering of conjunctive goals is activated [Biundo and Dengler, 1994] which decides to achieve the subgoal

[12]Note that $\neg open\_flag(mbox) = T$ is equivalent to $open\_flag(mailbox) = F$.

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

before the subgoal

$$delete\_flag(msg(x, mbox)) = T$$

because the former is a necessary precondition for an action achieving the latter. The first subgoal is isolated with the help of the *sel* rule [Biundo and Dengler, 1994]:

$$\frac{pre, \mathsf{Plan_A} \Rightarrow \Diamond[goal_1 \wedge \circ F \wedge pre'] \qquad pre', \mathsf{Plan_B} \Rightarrow \Diamond[goal_1 \wedge goal_2]}{pre, \mathsf{Plan_A} ; \mathsf{Plan_B} \Rightarrow \Diamond[goal_1 \wedge goal_2]}\, sel$$

This rule requires a sequential composition of two planvariables that can be split such that the first planvariable represents a subplan achieving the first subgoal, while the second planvariable represents a subplan achieving the remaining subgoals. But the planvariable $\mathsf{Plan_2}$ introduced by the *effect_split* rule has been instantiated with the action $ex(type(x, mbox))$ in specification 10. Thus, this instantiation must be withdrawn and plan refitting sets $\mathsf{Plan_2}$ to $\mathsf{Plan_5} ; \mathsf{Plan_6}$.

The first subgoal $read\_flag(msg(x, mbox)) = T$ has successfully been proven during plan interpretation using the old subgoal $read\_flag(msg(m, mybox)) = T$. Consequently, the action from the plan skeleton $ex(type(x, mbox))$ achieving the isolated subgoal is reused as an instantiation of the planvariable $\mathsf{Plan_5}$:

$$\begin{aligned} &open\_flag(msg(mbox)) = T \wedge delete\_flag(msg(x, mbox)) = F, \\ &ex(type(x, mbox)) \\ &\Rightarrow \Diamond[read\_flag(msg(x, mbox)) = T \wedge \circ F \wedge pre'] \end{aligned} \tag{13}$$

The instantiation can be successfully verified by plan refitting. The variable pre' is instantiated by computing frame conditions using PHI. With the second subgoal $delete\_flag(msg(x, mbox)) = T$ plan refitting addresses the open subgoal that plan interpretation failed to prove. Plan refitting concludes that the reused plan provides no instantiation and relies on planning from scratch. It generates the action $ex(delete(x, mbox))$ that instantiates the remaining planvariable $\mathsf{Plan_6}$. With this, all planvariables have been successfully instantiated and a correct proof of the plan specification has been constructed by plan refitting. The result is the desired plan **P2** that is obtained by reusing the sequential plan **P1**:

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

The planning process benefits from the reuse of plan **P1** in two situations:

- When a conditional control structure has to be introduced; here planning from second principles "knows" on which formula the case analysis has to be performed.
- When the subgoal $read\_flag(msg(m, mybox)) = T$ has to be addressed; here planning from second principles reuses an action instantiation that achieved the same goal in the plan candidate.

The search space during planning can be dynamically restricted in both cases, which leads to a speed up of the second-principles planner when compared to the generative planner.[13] A maximal reuse of the library plan is not possible according to the complexity-theoretic results from [Nebel and Koehler, 1993a]. In the example, this leads to some overhead during plan refitting where the action instance $ex(delete(m, mybox))$ is eliminated from the original plan, but subsequently re-introduced as the action instance $ex(delete(x, mbox))$. This demonstrates "that it is not possible to determine efficiently (i.e., in polynomial time) a maximal reusable plan skeleton *before* plan generation starts to extend this skeleton" (cf. [Nebel and Koehler, 1993a], page 1440).

The example demonstrated the generation of a conditional plan by reusing a sequential plan. MRL is the first system that is able to reuse and modify correctly plans containing control structures.

6.3 Reuse of Control Structures

The reuse and modification of plans with control structures leads to qualitatively new problems that do not occur in approaches restricting themselves to sequential plans. The modification of sequential plans comprises operations like the instantiation, deletion, addition or reordering of atomic actions. The modification of complex plans raises the question of whether these operations can be extended to control structures. Two main decisions have to be made:

1. Are control structures reused?
   versus
   Are only those sequential subplans reused that occur in the scope of control structures?
2. Are control structures introduced by the modification strategy if this is required by the refitting process?

[13] A summary of the results of an empirical study can be found in [Koehler, 1994d; Koehler, 1994b].

versus

Are control structures introduced if the current planning problem requires a plan containing control structures?

The treatment of control structures in a second-principles planner requires to make these decisions carefully and to take into consideration specific requirements from the application domain. The MRL system provides the reuse component of the PHI planner which is working in a help-system application. Here, plans are generated to provide active help to users of complex software environments [Bauer *et al.*, 1993]. This means that plans are required to meet exactly the user's goals and to be as simple as possible. Therefore, control structures are only reused in a restricted way in the implemented system MRL. They are introduced into the modified plan or preserved in the plan skeleton only if the current planning problem requires the generation of a plan containing control structures.

An unrestricted reuse of control structures can lead to the following problems:

- Reused control structures are not guaranteed to correspond to the requirements of the current planning situation. This can result in *over-complicated plans.* For example, a case analysis makes the execution of a plan more complicated because a test on the conditional has to be performed during execution time. Thus, a case analysis should only be introduced into a plan skeleton when the current planning problem requires us to generate a conditional plan.
- Plans can achieve *unintended side-effects.* Plan refitting makes some attempts at optimizing a reused plan by removing superfluous actions from it, but it is not able to generate optimal plans because this is usually harder than planning. Superfluous control structures render the problem worse. For example, an iterative plan which achieves a particular goal for all objects satisfying a precondition could in principle be reused to satisfy the goal for only one of the objects. As an example, the reader may think of reusing a plan achieving the goal "delete all my files in directory x" that achieves also the goal " delete file $x.ps$ in directory x". Without any attempts at optimizing the reused plan by removing the superfluous iterative control structure a drastic and harmful side effect is achieved.

Restricting the reuse of control structures as in MRL is one way of coping with these problems. Further research is necessary in order to identify other solutions.

7 Updating the Plan Library

The plan library is updated dynamically in MRL. The system starts with the initial plan library containing only the indices *top* and *bottom*. A new plan entry is added to the library under the following conditions:

- no reusable plan has been found and the planner has to generate a plan from first principles,
- the reused plan has to be modified.

The plan library is not updated when

- a library plan directly solves a current planning problem,
- the index of the current planning problem is already contained in the library.

MRL automatically builds a taxonomy of planning problems based on the indexing of plan specifications with the help of the encoding scheme. Each index represents an abstract class of planning problems in the application domain. An index is related to a *plan entry* containing stored information about a successfully solved planning problem: the plan, the plan specification, and information extracted from the planning process that has led to this plan. The plan in the plan entry is stored on a "first come—first serve" basis and represents an instance of the abstract class represented by the index. Planning problems belonging to the same abstract class can be solved by a modification of the stored plan in the plan entry. Thus, the plan library can be kept small. Furthermore, regularities of the application domain, e.g., typically occurring planning problems are reflected in the structure of the taxonomy.

Let us continue the example from Section 6. According to the above mentioned conditions, the plan library is updated because the reused plan has been modified. Three sources of information are available for the construction of the plan entry:

1. the current plan specification,
2. the modified plan satisfying the specification,
3. the proof tree that is stored as result of plan refitting.

The index of the plan entry has already been computed during plan determination. The current plan specification is completed before it is added to the

plan entry. This requires us first, to complete the goal specification, i.e., to specify additional goals a plan can achieve as side-effects and secondly, to minimize the preconditions of a plan, i.e., to eliminate preconditions from the specification formula which are not necessary for the plan.

The completion process analyses the instances of planning rules and action axiom schemata that have been applied during the proof performed by plan refitting. Action axiom schemata specify the necessary preconditions of an action and the effects it achieves (cf. Section 4). The computation of the index is repeated if the completion process leads to a changed specification formula.

In the example under consideration, the completion of plan specification S_{P2} leads to a disjunctive precondition reflecting the complete case analysis that has been introduced into the plan with the help of the *if_intro* rule:

$$
\begin{aligned}
&\mathsf{Plan}_{P2} \wedge \\
&[\mathit{delete\_flag}(\mathit{msg}(x, \mathit{mbox})) = F \wedge \mathit{open\_flag}(\mathit{mbox}) = T] \vee \\
&[\mathit{delete\_flag}(\mathit{msg}(x, \mathit{mbox})) = F \wedge \mathit{open\_flag}(\mathit{mbox}) = F] \\
&\rightarrow \Diamond [\mathit{read\_flag}(\mathit{msg}(x, \mathit{mbox})) = T \wedge \\
&\qquad \mathit{delete\_flag}(\mathit{msg}(x, \mathit{mbox})) = T]
\end{aligned}
$$

An explicit representation of the possible preconditions for plan **P2** supports the identification of applicable subplans during the plan interpretation phase. A recomputation of the index is not necessary because the conjunctive normal form of the completed precondition formula is logically equivalent to the originally specified precondition in S_{P2}.

A major part of a plan entry comprises information that is extracted from the proof tree leading to a plan:

- relation of sequential subplans occurring in conditional plans to their minimal preconditions,
- extraction of sequential body plans occurring in iterative plans,
- relation of atomic actions to the atomic goals achieved by the plan.

In order to relate sequential subplans to their minimal preconditions the proof tree is analyzed for applications of the rule *if_intro* (cf. Section 6). In the example, plan refitting has led to the conditional plan

$$
\begin{aligned}
\mathsf{Plan}_1 : \ &\textbf{if } \mathit{open\_flag}(\mathit{mbox}) = T \textbf{ then } \mathit{ex}(\mathit{empty\_action}) \\
&\qquad\qquad\qquad\quad \textbf{else } \mathit{ex}(\mathit{mail}(\mathit{mbox}))
\end{aligned}
$$

preceding the sequential plan $ex(type(x, mbox)) ; ex(delete(x, mbox))$. Two possible preconditions for this plan are explicitly represented in the completed plan specification formula. Now, each of them is related to the sequential plan that belongs to one of the preconditions. Consequently, the following information is stored in so-called *belongs_to* entries:

- $belongs\_to[ex(type(x, mbox)) ; ex(delete(x, mbox)),$
  $delete\_flag(msg(x, mbox)) = F \wedge open\_flag(mbox) = T]$
- $belongs\_to[ex(mail(mbox)) ; ex(type(x, mbox)) ; ex(delete(x, mbox)),$
  $delete\_flag(msg(x, mbox)) = F \wedge open\_flag(mbox) = F]$

Plan refitting relies furthermore on information about the relationship between actions and atomic subgoals. When a current atomic subgoal has successfully been proved with the help of an old subgoal during plan interpretation, plan refitting looks this old subgoal up in so-called *reaches* entries stored with the plan entry and reuses the action or subplan which achieved the old subgoal.

The action instances which achieve atomic goals are extracted from the leaves of the proof tree resulting from the application of action axiom schemata. In the example, we obtain the following *reaches* entries:

- $reaches[ex(mail(mbox)), open\_flag(mbox) = T]$
- $reaches[ex(type(x, mbox)), read\_flag(x, mbox) = T]$
- $reaches[ex(delete(x, mbox)), delete\_flag(msg(x, mbox)) = T]$

The construction of a plan entry is completed by a systematic renaming of variables with internal designators and by a sort-preserving abstraction of constants like sender *Joe* with existentially quantified variables.

Finally, the plan entry is related to its index which uniquely determines its position in the plan library. It is now available to subsequent planning from second principles.

8 Related Work

The implementation of a second-principles planner based on the formal framework as introduced in Section 3 requires design decisions that specify how planning from second principles proceeds in detail. In this section, we discuss the most important of these decisions underlying MRL and relate the system to other approaches.

Meta Level versus Object Level

Planning from second principles can proceed on a *meta level* or on an *object level*. On the *object level*, previously generated plans are directly reused to solve the current planning problem. This means that the plans as the objects of the planning process guide planning from second principles. On the *meta level*, knowledge extracted from previous planning processes representing "planning experience" guides the search for the desired plan.

The commitment of a particular planner to one of these levels is a fundamental design decision. A commitment to the object level leads to case-based planners and reuse systems, e.g., PRIAR [Kambhampati and Hendler, 1989] and SPA [Hanks and Weld, 1992a]. A commitment to the meta level leads to adaptive and reactive systems based on learning techniques, e.g., PRODIGY [Minton, 1988] and GRASSHOPPER [Leckie and Zuckerman, 1993].

MRL proceeds mainly on the object level because it relies on the reuse of stored plans. Meta level knowledge is reused, e.g., when plan refitting is supplied with information about preconditions on which case analyses have to be performed, cf. the example in Section 6.

Skeletal Plan Refinement versus Flexible Modification

If planning from second principles proceeds on the object level, plans are modified in order to construct the desired plan from them. Plan modification can be implemented as *skeletal plan refinement* [Friedland and Iwasaki, 1985] or as *flexible modification* [Kambhampati, 1990; Hanks and Weld, 1992a].

Skeletal plan refinement computes an appropriate ground-level instantiation for each operator occurring in the abstract skeleton. The admissible modification operations are restricted to *instantiation*, but they can proceed in several hierarchical steps and backtracking may occur. The modified plan is obtained as an instance of the skeleton.

Flexible modification as implemented in MRL admits a variety of operations on plans, e.g., the *deletion* and *addition* of operators and control structures.

Skeletal refinement occurs in MRL when the current plan specification has successfully been proved to be a logical instance of the reused plan specification. In this situation, an instance of the library plan will solve the current planning problem and plan modification can be restricted to instantiation operations.

Transformation-based versus Generation-based

The modification of a plan can be done with the help of transformations [McDermott, 1978; Hammond, 1990; Beetz and McDermott, 1992] or by extend-

ing a first-principles planner with the ability to modify plans [Kambhampati, 1989; Hanks and Weld, 1992a; Veloso, 1992b].

Transformation-based approaches execute a plan in a simulated environment. Failures are classified in a failure hierarchy and resolved by activating transformations on the plan. This approach requires a prediction of all possible failures, i.e., a proof of the *completeness* of the failure hierarchy and the available transformation rules, which is hard to achieve.[14] Further problems are related to the *soundness* and *termination* of the transformations. Transformations resolving a failure may introduce other failures, which makes it difficult to ensure that the transformation process does not loop and that the transformed plan is sound, i.e., that it solves the current planning problem.

To overcome these problems, a generation-based approach has been introduced in the PRIAR system [Kambhampati, 1989]. The proof of the completeness of plan modification with respect to the planner is trivial since plan modification can rely on plan generation as a "fall-back" possibility. Soundness and termination are also easy to ensure if the underlying first-principles planner possesses these properties.

The modification of a plan in MRL proceeds *generation-based.* MRL computes a *plan skeleton* and sends it to plan refitting for completion, which interacts with the generative PHI planner. The plan skeleton preserves those control structures and actions that are assumed to be reusable. The extension of a skeleton to a correct plan requires *flexible* modification operations, which *add* or *delete* new operators and control structures. The correctness of the planning process with the help of which the skeleton is completed, ensures that the modified plan is *sound.* Planning knowledge represented by the plan skeleton guides the current planning process and dynamically constrains the search space.

MRL is "complete" with respect to the planner because plan refitting can "fall back" on plan generation. The system is incomplete in the sense that it will not always find a plan if there is one because the use of tactics makes the underlying LLP theorem prover incomplete.

Conservative versus Non-Conservative

A desirable property of plan modification is *conservatism*, which means to "produce a plan ... by minimally modifying [the original plan]" [Kambhampati and Hendler, 1992]. Minimal modification of a plan implies a preservation the maximal number of applicable operators in a plan skeleton. A critical analysis of conservatism in [Nebel and Koehler, 1993a] shows that the computation of such maximal plan skeletons is PSPACE-hard. There-

[14] As an example see the incompleteness proof of CHEF in [Hanks and Weld, 1992b].

fore, implemented systems including MRL are non-conservative. In order to ensure efficiency of the plan modification process they rely on polynomial approximations, for example proof tactics for plan interpretation that run in polynomial time, which compute an "entry point" into the search space of possible plans as made explicit by Hanks and Weld [Hanks and Weld, 1992a]. This entry point cannot be guaranteed to be the best, but it is the best the approximation algorithm can compute. It is an open problem whether the maximal applicable subplan is approximable within a constant ratio. Recent results for similar problems [Bäckström, 1994] seem to hint at a negative result.

The Plan Library

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

One such extension is the system RAT [Heinsohn *et al.*, 1991] which is based on $\mathcal{KRIS}$ [Baader *et al.*, 1992]. RAT implements reasoning about plans by inferences in the underlying description 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 description 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.

The plan library can be *static* as well as *dynamic* in MRL. 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 main advantage in using a description logic as a query language to the plan library as in MRL lies in the theoretically well-founded properties of the retrieval algorithm. For the first time, retrieval is guaranteed to retrieve *solutions* from a library in *polynomial* time. This contrasts with approaches that are restricted to retrieving "reasonable similar past cases ... within limited bounded resources" (cf. [Veloso, 1992b], p. 103). Furthermore, an indexing of plan libraries based on the lattice structure provided by the *subsumption* hierarchy overcomes problems occurring in indexing schemes based on discrimination networks. On the one hand, discrimination networks fail in indexing complex plan specifications because they are restricted to coping with conjunctions of literals. On the other hand, such pathological situations

may occur where retrieval algorithms working on discrimination networks are provided with an exponentially growing input set. For example, given a goal state containing n atomic subgoals, the retrieval algorithm developed in [Veloso, 1992b] first searches a plan covering these n subgoals. If this fails, it computes all subsets of subgoals of cardinality $n-1$, then $n-2$ and so on until it takes the atomic subgoals as input. This means, the retrieval algorithm takes the power set of n except the empty set as input in the worst case, which is 2^n-1. An indexing based on the subsumption hierarchy avoids such problems because existing relationships between sets of subgoals are reflected in the lattice structure of the index taxonomy.

9 Conclusion

We have presented a logic-based approach to planning from second principles, which relies on a systematic decomposition of the planning process with the help of a four-phase model. Deductive inference processes with clearly defined semantics formalize each phase. The formal model is independent of a particular planning formalism and makes no commitments to application domains, implementational details, the nature of plans or planning situations.

Plan modification yields provably correct modified plans and enables a second-principles planner to reuse plans containing control structures like conditionals and iterations.

Reusable plans are retrieved from a dynamically updated plan library using a description logic as query language to the library. The plan library can be indexed based on a lattice structure and retrieval is formalized using a KL-ONE like classifier which is guaranteed to find existing solutions.

The formal framework leads to an implemented system with predictable behavior. Furthermore, in contrast to heuristic approaches, theoretical properties like the correctness, completeness and efficiency of the inference procedures can be proved.

Acknowledgements

This work was supported by the German Ministry for Research and Technology (BMFT) under contract ITW 9000 8 as part of the PHI project. I want to thank the members of the PHI group, Mathias Bauer, Susanne Biundo, Dietmar Dengler, and Gaby Paul for their interest in my work and for fruitful discussions. Furthermore, I am indebted to Wolfgang Wahlster for his advice and support and to Bernhard Nebel for joint work. Hans-Jürgen Ohlbach made helpful comments on an earlier version of this paper.

References

[AIPS-92, 1992] *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, Washington, D.C., 1992. Morgan Kaufmann, San Mateo.

[Baader *et al.*, 1992] 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.

[Bäckström and Sandewall, 1994] C. Bäckström and E. Sandewall, editors. *Current Trends in AI Planning.* IOS Press, Amsterdam, Washington, Tokyo, 1994.

[Bäckström, 1994] C. Bäckström. Finding least constrained plans and optimal parallel executions is harder than we thought. In Bäckström and Sandewall [1994].

[Bauer and Paul, 1994] M. Bauer and G. Paul. Logic-based plan recognition for intelligent help systems. In Bäckström and Sandewall [1994], pages 60–73.

[Bauer *et al.*, 1993] M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul. PHI - a logic-based tool for intelligent help systems. In IJCAI-93 [1993], pages 460–466.

[Beetz and McDermott, 1992] M. Beetz and D. McDermott. Declarative goals in reactive plans. In AIPS-92 [1992], pages 3–12.

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

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

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

[Constable, 1986] R. Constable. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[Feigenbaum, 1963] 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.

[Friedland and Iwasaki, 1985] P. Friedland and Y. Iwasaki. The concept and implementation of skeletal plans. *Journal of Automated Reasoning*, 1:161–208, 1985.

[Frisch and Scherl, 1991] A. M. Frisch and R. B. Scherl. A general framework for modal deduction. In J.A. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 196–207. Morgan Kaufmann, San Mateo, 1991.

[Gallier, 1987] J. H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. John Wiley and Sons, New York, 1987.

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

[Green, 1969] C. Green. Application of theorem proving to problem solving. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 219–239, Washington, D.C., May 1969.

[Hammond, 1990] K. J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45:173 - 228, 1990.

[Hanks and Weld, 1992a] S. Hanks and D. Weld. Systematic adaptation for case-based planning. In AIPS-92 [1992], pages 96–105.

[Hanks and Weld, 1992b] S. Hanks and D. Weld. The systematic plan adaptor: A formal foundation of case-based planning. Technical Report 92-09-04, University of Washington, Department of Computer Science and Engineering, 1992.

[Heinsohn *et al.*, 1991] 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.

[Heisel *et al.*, 1990] M. Heisel, W. Reif, and W. Stephan. Tactical theorem proving in program verification. In *Proceedings of the 10th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 449, pages 117–131, Kaiserslautern, Germany, 1990. Springer, Berlin.

[Hollunder *et al.*, 1990] 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, 1993] *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, August 1993. Morgan Kaufmann.

[Kambhampati and Hendler, 1989] S. Kambhampati and J.A. Hendler. Control of refitting during plan reuse. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 943–949, Detroit, MI, August 1989. Morgan Kaufmann.

[Kambhampati and Hendler, 1992] S. Kambhampati and J.A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55:193 – 258, 1992.

[Kambhampati, 1989] S. Kambhampati. Flexible reuse and modification in hierarchical planning: A validation-structure-based approach. PhD Thesis MD 207 42-3411, University of Maryland, Center for Automation Research, Computer Vision Laboratory, 1989.

[Kambhampati, 1990] S. Kambhampati. A theory of plan modification. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 176–182, Boston, MA, August 1990. MIT Press.

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

[Koehler and Treinen, 1993] 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.

[Koehler, 1994a] J. Koehler. An application of terminological logics to case-based reasoning. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 351–362. Morgan Kaufmann, San Francisco, CA, 1994.

[Koehler, 1994b] J. Koehler. Avoiding pitfalls in case-based planning. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, 1994. Morgan Kaufmann, San Mateo.

[Koehler, 1994c] J. Koehler. Correct modification of complex plans. In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 605–609, Amsterdam, NL, August 1994. John Wiley & Sons.

[Koehler, 1994d] J. Koehler. Flexible plan reuse in a formal framework. In Bäckström and Sandewall [1994], pages 171–184.

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

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

[Kowalski, 1979] R. Kowalski. *Logic for Problem Solving.* North Holland, Amsterdam, 1979.

[Leckie and Zuckerman, 1993] C. Leckie and I. Zuckerman. An inductive approach to learning search control rules for planning. In IJCAI-93 [1993], pages 1100–1105.

[Manna and Waldinger, 1987a] Z. Manna and R. Waldinger. How to clear a block: Plan formation in situational logic. *Journal of Automated Reasoning*, 3:343–377, 1987.

[Manna and Waldinger, 1987b] Z. Manna and R. Waldinger. A theory of plans. In M. Georgeff and A. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pages 11–45. Morgan Kaufmann, 1987.

[McDermott, 1978] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[Minton, 1988] S. Minton. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 564–569, Saint Paul, MI, August 1988. MIT Press.

[Nebel and Koehler, 1993a] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In IJCAI-93 [1993], pages 1436–1441.

[Nebel and Koehler, 1993b] 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.

[Nebel *et al.*, 1992] B. Nebel, W. Swartout, and C. Rich, editors. *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, San Mateo, 1992.

[Nebel, 1990] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence 422. Springer, 1990.

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

[Ohlbach, 1991] H.J. Ohlbach. Semantics-based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–775, 1991.

[Paulson, 1990] L. Paulson. Isabelle: The next 700 theorem provers. In P. Odifredi, editor, *Logic and Computer Science.* Academic Press, 1990.

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

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

[Veloso, 1992a] M. Veloso. Automatic storage, retrieval, and replay of multiple cases using derivational analogy in PRODIGY. In *Working Notes of the AAAI Spring Symposium on Computational Considerations in Supporting Incremental Modification and Reuse*, pages 131–136, Stanford University, CA, 1992.

[Veloso, 1992b] M. Veloso. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, Carnegie Mellon University, 1992.

[Wallen, 1989] L.A. Wallen. *Automated Deduction in Nonclassical Logics.* MIT Press, Cambridge, London, 1989.

[Weida and Litman, 1994] 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.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or via anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.

The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-93-09
Philipp Hanschke, Jörg Würtz:
Satisfiability of the Smallest Binary Program
8 pages

RR-93-10
Martin Buchheit, Francesco M. Donini, Andrea Schaerf: Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-11
Bernhard Nebel, Hans-Jürgen Bürckert:
Reasoning about Temporal Relations:
A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-12
Pierre Sablayrolles: A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-13
Franz Baader, Karl Schlechta:
A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen:
Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster: PLUS - Plan-based User Support
Final Project Report
33 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-17
Rolf Backofen:
Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder:
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller:
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy:
Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel:
Document Highlighting —
Message Classification in Printed Business Letters
17 pages

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-27
Hans-Ulrich Krieger:
Derivation Without Lexical Rules
33 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker: Feature-Based Allomorphy
8 pages

RR-93-29
Armin Laux: Representing Belief in Multi-Agent Worlds viaTerminological Logics
35 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman:
Corporate Agents
14 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman:
Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman:
Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-33
Bernhard Nebel, Jana Koehler:
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster:
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.):
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf:
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen:
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-43
M. Bauer, G. Paul: Logic-based Plan Recognition for Intelligent Help Systems
15 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt: Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-45
Rainer Hoch: On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-46
Philipp Hanschke: A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder:
Cardinality Restrictions on Concepts
20 pages

RR-94-01
Elisabeth André, Thomas Rist:
Multimedia Presentations:
The Support of Passive and Active Viewing
15 pages

RR-94-02
Elisabeth André, Thomas Rist:
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-03
Gert Smolka:
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-05
Franz Schmalhofer, J.Stuart Aitken, Lyle E. Bourne jr.:
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-06
Dietmar Dengler:
An Adaptive Deductive Planning System
17 pages

RR-94-07
Harold Boley: Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-08
Otto Kühn, Björn Höfling: Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-10
Knut Hinkelmann, Helge Hintze:
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-11
Knut Hinkelmann: A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-12
Hubert Comon, Ralf Treinen:
Ordering Constraints on Trees
34 pages

RR-94-13
Jana Koehler: Planning from Second Principles —A Logic-based Approach
49 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr: Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-16
Gert Smolka: A Foundation for Higher-order Concurrent Constraint Programming
26 pages

DFKI Technical Memos

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben:
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

TM-93-01
Otto Kühn, Andreas Birk: Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

TM-93-02
Pierre Sablayrolles, Achim Schupeta:
Conlfict Resolving Negotiation for COoperative Schedule Management
21 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

TM-93-04
Hans-Günther Hein:
Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages

TM-93-05
Michael Sintek: Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores:
Text Skimming as a Part in Paper Document Understanding
14 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll:
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

DFKI Documents

D-93-07
Klaus-Peter Gores, Rainer Bleisinger:
Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-08
Thomas Kieninger, Rainer Hoch:
Ein Generator mit Anfragesystem für strukturierte Wörterbücher zur Unterstützung von Texterkennung und Textanalyse
125 Seiten

D-93-09
Hans-Ulrich Krieger, Ulrich Schäfer:
TDL ExtraLight User's Guide
35 pages

D-93-10
Elizabeth Hinkelman, Markus Vonerden, Christoph Jung: Natural Language Software Registry (Second Edition)
174 pages

D-93-11
Knut Hinkelmann, Armin Laux (Eds.):
DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

D-93-12
Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek, Werner Stein:
RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-14
Manfred Meyer (Ed.): Constraint Processing – Proceedings of the International Workshop at CSAM'93, July 20-21, 1993
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-15
Robert Laux:
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt: Design & KI
74 Seiten

D-93-20
Bernhard Herbig:
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-21
Dennis Drollinger:
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-22
Andreas Abecker:
Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

D-93-24
Brigitte Krenn, Martin Volk:
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
66 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.):
Modeling Epistemic Propositions
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-26
Frank Peters: Unterstützung des Experten bei der Formalisierung von Textwissen
INFOCOM:
Eine interaktive Formalisierungskomponente
58 Seiten

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkoreit (Eds.):
Report of theEAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-94-01
Josua Boon (Ed.):
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

D-94-02
Markus Steffens: Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-06
Ulrich Buhrmann:
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 pages

D-94-08
Harald Feibel: IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

Planning from Second Principles—A Logic-based Approach

Jana Koehler

RR-94-13
Research Report