Flexible Plan Reuse in a Formal Framework

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

Abstract. The reuse of plans is widely considered as a valuable tool for the improvement of efficiency in planning systems because it avoids the repetition of planning effort. Several approaches which investigate the modification and reuse of sequential plans in the framework of STRIPS-based planning have been developed.
In this paper we present a domain-independent approach to flexible plan reuse based on a deductive framework. A formalization of the whole reuse process is proposed including the modification, representation and retrieval of plans.
Plan modification is based on a semantic approach which yields provably correct modified plans. The plan library is represented in a hybrid knowledge representation formalism linking the planning logic with a terminological logic and is dynamically created and maintained by the system requiring no user interaction.
Apart from sequential plans, this approach enables a planner to reuse and modify plans containing control structures like conditionals and iterations.
Comparing the performance of the deductive plan reuse system with performance measures reported from other systems it turns out that deductive approaches can compete very well with heuristic ones.

1. Introduction

The reuse and modification of previously synthesized artifacts such as plans, designs, proofs and programs to meet new specifications is considered as a valuable tool for the improvement of efficiency in many tasks. Consequently, reuse and modification has emerged as an active topic of research in may areas of AI, including *planning*, *scheduling*, *design*, *automated theorem proving* and *software engineering*.
Current research revolves around understanding the spectrum of reuse and modification tasks, the usefulness of various techniques and their formalization.
The introduction of reuse and modification techniques into the field of planning originated from research in *case-based reasoning* and *robotics*. The work was mainly motivated

*This paper has been published in *Current Trends in AI Planning*, ed. by C. Bäckström and E. Sandewall, pages 171-184, IOS Press, Amsterdam, Washington, Tokyo.

by flexibility and efficiency gains expected by researchers when planners are provided with the ability to modify and reuse plans in the case of execution failures or changes of planning problem specifications. Current research in case-based planning led to a wide spectrum of approaches (cf. for a comprehensive overview [5, 20, 6, 21]). While there exist general frameworks extending STRIPS-based planners, e.g. the PRIAR system [12, 13] and the SPA system [10, 9], there are no known approaches which study plan reuse in the context of *deductive planning*.

Planning can be seen as an inference process. Starting from the formal description of a start situation and a set of available operators, it has to be proven how a situation can be achieved that is sufficient for the formally described goal. This view led to the field of *deductive planning* [8, 15, 2, 16] where plans are generated by an inference process in an underlying logic.

The system MRL[1] we present in this paper, integrates a plan reuse facility into a deductive planner. Plan reuse is based on a logical formalization of the reuse process including the modification, representation and retrieval of plans. The system is able to automatically reuse and modify sequential, conditional, and iterative plans. The plan library is represented in a hybrid knowledge representation formalism linking the planning logic with a terminological logic. It is dynamically created and maintained by the system and therefore requires no user interaction [14].

The main emphasis of the paper is on plan modification. Section 2 introduces a four phase model as a basis for the formalization of the plan reuse process and describes how plan modification and plan-library operations are formalized. The deductive approach to plan modification is presented in section 3. The section shortly reviews the planner into which the reuse facility is integrated and gives a detailed example. In section 4 we compare performance measures reported by various reuse systems and conclude with some final remarks in section 5.

2. Plan Reuse in a Logical Framework

Following a logical approach, plan reuse leads to modified plans which are provably correct. Furthermore, since plan modification is done deductively, a semantic comparison of planning problems is possible instead of a syntactic match. A domain-independent formalism which possesses a clearly defined semantics is obtained and the formal view on plan reuse facilitates the investigation of its theoretical properties.

2.1. A four Phase Model of Plan Reuse

The process of reusing previously generated plans in order to solve new planning problems proceeds in the following steps:

(I) Plan Determination: After the deductive planning system has received a formal plan specification, the plan reuse process starts by searching in a plan library. A plan library contains a collection of *plan entries* which 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

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

extracted from the plan generation process. The current plan specification is used as a search key in order to determine an appropriate plan entry containing a "similar" plan specification and therefore a reusable plan.
(II) Plan Interpretation: In the phase of *plan interpretation* the current and the reused plan specification are compared semantically. As we propose a deductive approach to plan reuse, this semantic comparison is implemented as a theorem proving attempt. The result of the plan interpretation phase is a proof, which states that the plan belonging to the reused plan specification can be reused without modification, or a failed proof from which refitting information can be extracted.
(III) Plan Refitting: *Plan refitting* starts by constructing a *plan skeleton* from the reused plan in accordance with the result of the proof attempt. Plan skeletons are plan formulae in which plan metavariables occur as "placeholders" for subplans which must be found during plan refitting. Plan skeletons are *derived* from the reused plan according to a *modification strategy* which deletes actions and control structures from the plan which are not reused and adds metavariables at positions in the skeleton where newly generated subplans have to be introduced for open subgoals, which the reused plan is unable to reach. The plan skeleton keeps any actions of the reused plan which were determined as reusable during the proof attempt and in which variables are appropriately instantiated with object parameters taken from the current plan specification.
The plan skeleton is used as an instantiation for the plan metavariable which occurs in the current plan specification. It is *extended* to a correct plan by replacing each plan metavariable by a (possible empty) subplan during a constructive proof of the instantiated plan specification formula.
(IV) Plan-Library Update: The reuse process finishes with a *plan-library update*, in which a new plan entry is constructed from 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 extension of the plan skeleton.

The four phase model describes a temporal view of the reuse process. The phases I to III are necessary in order to generate a plan by reusing an existing one and are also distinguished by other authors who frequently denote them as *retrieval - matching - adaptation* phases. 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 phase I and IV and are only shortly discussed in this paper. Plan interpretation and refitting work on plan specifications and are summarized as *plan modification*, a subject which is described in detail in the following.

2.2. Formalization of Plan Modification

In deductive planning systems, planning problems are given as formal plan specifications, i.e. formulae in the underlying logic. Plan generation in these systems is done by a constructive proof of the specification formula leading to a plan that is *sufficient* for the plan specification. A plan Plan_{old}, which solves a specification Spec_{old} is a solution for another specification Spec_{new} if Spec_{new} is a logical instance of Spec_{old}. In this case, solving Spec_{old} is sufficient for solving Spec_{new}. Formally expressed, this means that a plan Plan_{old} can be reused if the proof

$$\boxed{\mathsf{Spec}_{old} \rightarrow \mathsf{Spec}_{new}}$$

is successful. A successful proof can be interpreted in such a way, that the current planning problem has been shown to be a logical instance of the reused planning problem. An instance of the reused plan will therefore also solve the current planning problem. If the proof fails, the plan Plan_{old} is modified based on information extracted from the failed proof.
The implementation of the proof attempt in a particular deductive planning system requires to specify the subproofs that have to performed in order to show that $\mathsf{Spec}_{old} \rightarrow \mathsf{Spec}_{new}$ is valid. In general, formal plan specifications comprise a description of the preconditions that hold in the initial state and a description of the goals the plan has to achieve. Therefore, the proof of the relation $\mathsf{Spec}_{old} \rightarrow \mathsf{Spec}_{new}$ can be reduced to proofs of relations between preconditions and goals: The plan Plan_{old} is a solution if

1. the current preconditions specified in Spec_{new} are sufficient for the preconditions the plan Plan_{old} requires, i.e. the reused plan is applicable in the current initial state. By proving the subgoal

   $$pre_{new} \rightarrow pre_{old}$$

   we show that at least the preconditions the plan requires hold in the current initial state.

2. the goals specified in Spec_{old} are sufficient for the current goals required in Spec_{new}, i.e. the reused plan achieves at least all of the current goals. By proving the subgoal

   $$goal_{old} \rightarrow goal_{new}$$

   we show that the plan achieves at least all goals that are required in Spec_{new}.

2.3. Formalization of the Plan Library

According to the formalization of plan modification the plan library has to be searched for a plan specification which logically implies the current one. Furthermore, the plan library should be structured based on the implication relation in order to facilitate the identification of a reusable plan. Unfortunately, the implication relation is undecidable in expressive formalisms like first order predicate logic. But the search criterion underlying the retrieval process should at least be decidable. Therefore, instead of using the implication relation directly, an approximation of $P_{old} \rightarrow P_{new}$ is developed. This is done with the help of an *encoding scheme* ω:

$$\boxed{\omega(\mathsf{Spec}_{old}) \rightarrow^{\omega} \omega(\mathsf{Spec}_{new})}$$

The encoding scheme of plan specifications leads to *indices* of plan entries in the library. Each plan entry possesses an index that has been constructed by the encoding of the plan specification. In MRL, terminological logics are used as the underlying representation formalism for indices: an index $\omega(Spec)$ is represented as a *concept* in a terminological logic. The relation $\rightarrow^{\omega}$ between indices is formalized as *subsumption* between concepts. With that, the retrieval of a similar plan from the plan library can be grounded on *concept classification* [14].

3. Deductive Plan Modification in PHI

The general approach presented in this paper provides the formal basis for the reuse component MRL. MRL has been developed as an integrated part of the system PHI, a logic-based tool for intelligent help systems which integrates plan generation and plan recognition components [4, 1]. The planner and the recognizer work in close mutual cooperation and rely on a common logical framework, i.e. the interval-based modal temporal logic LLP [3].
LLP provides the modal operators $\bigcirc$ (next), $\Diamond$ (sometimes), $\Box$ (always) and the binary modal operator ; (chop) which expresses the sequential composition of formulae. As in programming logics, control structures like iterations and conditionals and *local variables* the value of which may vary from state to state are available.
Plans are represented by a certain class of LLP formulae. They may contain, e.g. basic actions which are expressed by the *execute* predicate ex, the *chop* operator ; and control structures.
Plan specifications are LLP formulae of form $[\mathit{preconditions} \wedge \mathsf{Plan}] \rightarrow \mathit{goals}$, i.e. if the Plan is carried out in a situation where the *preconditions* hold then the *goals* will be reached.
Plan generation is done deductively by performing constructive proofs of plan specifications in a sequent calculus which was developed for LLP. During this proof, the plan metavariable Plan is replaced by a plan (formula) satisfying the specification. The proofs are guided by tactics which can be described in a tactic language provided by the system, an idea which was borrowed from the field of *tactical theorem proving* [7, 11, 19]. 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 of a manageable size and only those deduction steps which appear to be the most promising are performed.
The application domain of PHI is the UNIX *mail domain* where objects like *messages* and *mailboxes* are manipulated by actions like *read*, *delete*, and *save*.
The atomic actions available to the planner are the elementary commands of the UNIX mail system. They are axiomatized like assignment statements in programming logics. State changes which are caused by 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 $mbox$ reads

$$\begin{aligned} \forall x\, open\_flag(mbox) = T \wedge delete\_flag(msg(x, mbox)) = F \wedge ex(delete(x, mbox)) \\ \rightarrow \bigcirc delete\_flag((msg(x, mbox)) = T \end{aligned}$$

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.

3.1. Plan Interpretation

Assume that the following plan specification formula Spec_{new}

$$
\begin{array}{l}
[open\_flag(M) = T \wedge delete\_flag(msg(x,M)) = F] \vee \\
[open\_flag(M) = T \wedge delete\_flag(msg(x,M)) = T] \wedge \mathsf{Plan}_{new} \\
\rightarrow \Diamond[read\_flag(msg(x,M)) = T \wedge \Diamond[save\_file(msg(x,M)) = file \wedge \\
\qquad \Diamond[delete\_flag(msg(x,M)) = T\,]\,]\,]
\end{array}
$$

has been forwarded from the planner to the reuse component. The specification describes the goal "Read, save, and then delete message x in mailbox M". In contrast to STRIPS-based planners, a temporal ordering of subgoals can be specified with the help of the $\Diamond$-Operator: the specification requires that the goal "read the mail" is achieved first, then the goal "save the mail" is achieved, followed by the goal "delete the mail". In order to facilitate the understanding of the example, only one atomic subgoal is specified in each intermediate subgoal state, but the PHI system and its reuse component MRL are also able to handle conjunctive goals.
The goal has to be reached under two different possible preconditions expressing uncertain knowledge about the initial situation: the mailbox M is open, but we do not know whether the message x has already been deleted or not.
Assume that furthermore, the search in the plan library has been completed successfully and has led to the reused plan specification Spec_{old}

$$
\begin{array}{l}
[open\_flag(Y) = T \wedge delete\_flag(msg(X,Y)) = F] \vee \\
[open\_flag(Y) = T \wedge delete\_flag(msg(X,Y)) = T] \wedge \mathsf{Plan}_{old} \\
\rightarrow \Diamond[read\_flag(msg(X,Y)) = T \wedge \Diamond[delete\_flag(msg(X,Y)) = T\,]\,]
\end{array}
$$

which achieves a subset of the current goals under the same preconditions by executing the plan Plan_{old}

$$
\begin{array}{l}
\textbf{if } delete\_flag(msg(X,Y)) = F \textbf{ then } ex(empty\_action) \\
\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } ex(undelete(X,Y)) \ ; \\
ex(type(X,Y)) \ ; \ ex(delete(X,Y))
\end{array}
$$

The plan interpretation phase starts with the proof $\mathsf{Spec}_{old} \rightarrow \mathsf{Spec}_{new}$ and performs the two subproofs $pre_{new} \rightarrow pre_{old}$ and $goal_{old} \rightarrow goal_{new}$.
The relation between preconditions and goals could also be checked by syntactically comparing the state descriptions as it is done in most plan reuse systems. Performing a proof on the formulae which represent the states can be viewed as a semantic comparison. During a proof, knowledge concerning regularities in the planning domain in form of deduction rules can be applied, e.g. in the mail domain we can conclude that if the mailbox is not open, then all the messages in the mailbox have not yet been deleted.
In order to obtain an appropriate instantiation of the reused plan, variables in the reused specification Spec_{old} are substituted by terms which occur in the current specification Spec_{new} when unification operations are being performed during the proof.
In the example, the validity of the relation between the preconditions is obvious. The proof of the relation between the goals requires to prove the following sequent in the LLP sequent calculus:

$$
\begin{array}{l}
\Diamond[read\_flag(msg(X,Y)) = T \wedge \Diamond[delete\_flag(msg(X,Y)) = T\,]\,] \\
\Rightarrow \\
\Diamond[read\_flag(msg(x,M)) = T \wedge \Diamond[save\_file(msg(x,M)) = file \wedge \\
\qquad \Diamond[delete\_flag(msg(x,M)) = T\,]\,]\,]
\end{array}
\qquad (1)
$$

Special purpose proof tactics guide the proof attempt in the LLP sequent calculus. Each tactic describes a well defined ordering of deduction rule applications. If a proof attempt fails in one tactic, then another tactic is applied which uses another ordering of deduction rules. The proof tactics are designed in such a way so that they always terminate. In addition, they are considered as decision procedures: If no tactic resulted in a proof tree, it is assumed that no proof is possible and that a falsifying valuation for some of the leaves in each deduction tree has been obtained.
The tactic for the goal proof uses the following sequent rules:

- $\dfrac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta}\ l\wedge$
- $\dfrac{\Gamma \Rightarrow A, \Delta \qquad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta}\ r\wedge$
- $\dfrac{\Gamma^*, A \Rightarrow \Delta^*}{\Gamma, \Diamond A \Rightarrow \Delta}\ l\Diamond$[2]
- $\dfrac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow \Diamond A, \Delta}\ r\Diamond$

The tactic starts by applying the rules $l\Diamond$, $r\Diamond$, $l\wedge$, $r\wedge$ to sequent 1 and thus reaches sequent 2.

$$\begin{array}{l} read\_flag(msg(X,Y)) = T, \Diamond delete\_flag(msg(X,Y)) = T \\ \Rightarrow \\ read\_flag(msg(x,M)) = T, \\ \qquad \Diamond[save\_file(msg(x,M)) = file \wedge \Diamond[delete\_flag(msg(x,M)) = T\,]\,] \end{array} \tag{2}$$

Applying the substitutions $\{x/X, M/Y\}$ the following axiom can be extracted from this sequent:

(A1) $read\_flag(msg(x,M)) = T \Rightarrow read\_flag(msg(x,M)) = T$

The tactic repeats the application of the sequent rules $l\Diamond$, $r\Diamond$, $r\wedge$ reaching the sequent

$$\begin{array}{l} delete\_flag(msg(X,Y)) = T \\ \Rightarrow \\ save\_file(msg(x,M)) = file, \Diamond delete\_flag(msg(x,M)) = T \end{array} \tag{3}$$

In contrast to the first cycle of the tactic no axiom can be extracted from this sequent. Therefore, the tactics enters a subcycle during which the right side of the sequent is split further. Applying the rule $r\Diamond$ leads to the sequent

$$\begin{array}{l} delete\_flag(msg(X,Y)) = T \\ \Rightarrow save\_file(msg(x,M)) = file, delete\_flag(msg(x,M)) = T \end{array} \tag{4}$$

from which the axiom

(A2) $delete\_flag(msg(x,M)) = T \Rightarrow delete\_flag(msg(x,M)) = T$

is extracted.

[2]With Γ^* and Δ^* defined as follows: $\Gamma^* \stackrel{\text{df}}{=} \{\Box B | \Box B \in \Gamma\}$ and $\Delta^* \stackrel{\text{df}}{=} \{\Diamond B | \Diamond B \in \Delta\}$.

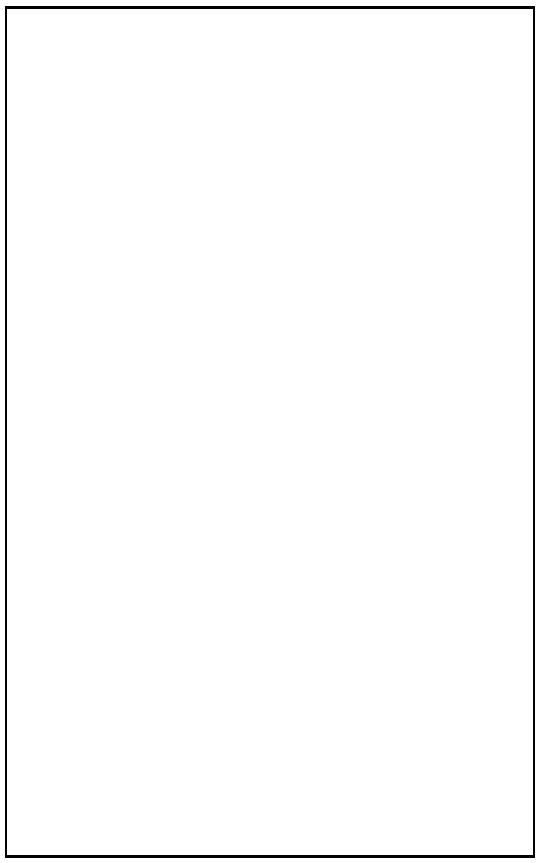


Figure 1: Proof of the goal relation $goal_{old} \rightarrow goal_{new}$ as visualized in MRL.

Figure 1 illustrates the deduction tree which is constructed by the tactic during the example proof. The black nodes designate the two axioms that were found. Both axioms represent the subgoals which occur in the current specification as well as in the reused specification, i.e. these subgoals of the current specification will be reached by the reused plan. The white node (treeID293) visualizes a leaf that could not be closed during the proof attempt, i.e. the tactic was unable to prove that the reused subplan achieves the current subgoal $save\_file(msg(x, M)) = file$. The white node (treeID296) represents a possible starting node of another deduction tree which was not extended further as the tactic terminated when this node was reached.
As one of the leaves of the sequent derivation is not an axiom, the tactic failed in proving that $goal_{old} \rightarrow goal_{new}$ holds. The plan does not achieve all of the required goals and thus plan refitting must begin.

3.2. Plan Refitting

After the termination of the proof tactics in the sequent calculus, several deduction trees are obtained as result of the plan interpretation phase. Two situations are now possible:

1. One of the deduction trees is a proof tree, 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 deduction tree is a proof tree and the assumption is made that no proof is possible. In this situation a set of counter-example trees was obtained where at least one falsifying valuation for some leaf in every deduction tree was found.

The leaves of a counter-example tree contain only atomic formulae which was guaranteed by the proof tactics. The falsifying valuation makes all atomic formulae describing goals from Spec_{old} true, however some of the atomic formulae which describe current goals from Spec_{new} are valued as false. These falsified goals are interpreted as those goals that are not achieved by the reused plan.
In the example, one deduction tree containing the non-axiom leaf (3) was constructed from which plan refitting concludes that the reused plan does not reach the current goal

$$save\_file(msg(x, M)) = file.$$

In general, plan refitting selects the smallest counter-example tree from the set of counter-example trees which were constructed during the plan interpretation phase. If no unique smallest counter-example exists, one of them is chosen arbitrarily. The selection of a minimal counter-example tree is based on a heuristic, which assumes that the smallest number of falsified current atomic goals leads to the minimal refitting effort.
In the example, the proof of the preconditions was successful; this means that the plan is applicable in the current initial state. But as the proof of the current goals failed, this plan must be modified by constructing a plan skeleton from it. After instantiating the reused plan with the substitutions computed during the proof attempt, a metavariable for the missing subgoal has to be introduced. In order to determine the position in the skeleton where this metavariable has to be added, the current goal state specification is analyzed with the help of the *effect_split* rule [4].

$$\frac{pre \wedge \mathsf{Plan}_1 \rightarrow \Diamond[goal_1 \wedge \bigcirc F \wedge pre_1] \qquad pre_1 \wedge \mathsf{Plan}_2 \rightarrow \Diamond goal_2}{pre \wedge \mathsf{Plan}_1 \,;\, \mathsf{Plan}_2 \rightarrow \Diamond[goal_1 \wedge \Diamond goal_2]}$$

The intention of this rule is to split a compound goal into the temporally first goal and the remaining goals. In parallel to this, a sequential plan is split into two temporally ordered subplans where the first subplan reaches the first goal and the second subplan reaches the remaining goals.
Since the preconditions in Spec_{new} are represented by a disjunction, a conditional control structure is introduced in the plan. The plan metavariable Plan_{new} is instantiated with

$$\begin{array}{lll} \textbf{if } delete\_flag(msg(x, M)) = F \wedge open\_flag(M) = T & \textbf{then} & \mathsf{Plan}_{new_1} \\ & \textbf{else} & \mathsf{Plan}_{new_2} \end{array}$$

The proof is split into two subproofs for each conditional. The **then** branch reads

$$\begin{array}{l} delete\_flag(msg(x, M)) = F \wedge open\_flag(M) = T \wedge \mathsf{Plan}_{new_1} \\ \Rightarrow \Diamond[read\_flag(msg(x, M)) = T \wedge \Diamond[save\_file(msg(x, M)) = file \wedge \\ \qquad \Diamond[delete\_flag(msg(x, M)) = T\,]\,]\,] \end{array} \tag{5}$$

The **else** branch reads

$$\begin{array}{l} delete\_flag(msg(x, M)) = T \wedge open\_flag(M) = T \wedge \mathsf{Plan}_{new_2} \\ \Rightarrow \Diamond[read\_flag(msg(x, M)) = T \wedge \Diamond[save\_file(msg(x, M)) = file \wedge \\ \qquad \Diamond[delete\_flag(msg(x, M)) = T\,]\,]\,] \end{array} \tag{6}$$

The analysis of the goal state is performed for each conditional. We show it for the **else** branch and begin with formula 6.
Plan refitting assumes that Plan_{new_2} is a sequential plan and introduces a sequential structure into it by substituting Plan_{new_2} with $\mathsf{Plan}_1 ; \mathsf{Plan}_2$

$$\begin{array}{l} delete\_flag(msg(x,M)) = T \wedge open\_flag(M) = T \wedge \mathsf{Plan}_1 ; \mathsf{Plan}_2 \\ \Rightarrow \Diamond[read\_flag(msg(x,M)) = T \wedge \\ \quad \Diamond[save\_file(msg(x,M)) = file \wedge \\ \quad\quad \Diamond[delete\_flag(msg(x,M)) = T]]] \end{array} \tag{7}$$

Now the *effect_split* rule is applied leading to the formulae 8 and 9:

$$\begin{array}{l} delete\_flag(msg(x,M)) = T \wedge open\_flag(M) = T \wedge \mathsf{Plan}_1 \\ \Rightarrow \Diamond[read\_flag(msg(x,M)) = T \wedge \circ F \wedge pre_1] \end{array} \tag{8}$$

$$\begin{array}{l} pre_1 \wedge \mathsf{Plan}_2 \\ \Rightarrow \Diamond[save\_file(msg(x,M)) = file \wedge \Diamond delete\_flag(msg(x,M)) = T] \end{array} \tag{9}$$

Subplan Plan_1 has to reach the first subgoal $read\_flag(msg(x,M)) = T$. This subgoal has been proven successfully in axiom (A1) using the subgoal $read\_flag(msg(X,Y)) = T$ from Spec_{old}. Consequently, the instantiated subplan reaching this subgoal in Plan_{old} is reused as an instantiation of the plan metavariable Plan_1:

$$ex(undelete(x,M)) ; ex(type(x,M))$$

Since formula 9 contains a compound goal, we assume that Plan_2 represents an action sequence, replace it by $\mathsf{Plan}_3 ; \mathsf{Plan}_4$ and then apply the *effect_split* rule again.

$$pre_1 \wedge \mathsf{Plan}_3 \Rightarrow \Diamond[save\_file(msg(x,M)) = file \wedge \circ F \wedge pre_2] \tag{10}$$

$$pre_2 \wedge \mathsf{Plan}_4 \Rightarrow \Diamond delete\_flag(msg(x,M)) = T \tag{11}$$

With formula 10 we isolated the open subgoal; the reused plan provides no instantiation for Plan_3, while Plan_4 is replaced by the action $ex(delete(x,M))$ which was identified as reusable. Performing the same analysis for the **then** branch of the conditional leads to the following plan skeleton

if $delete\_flag(msg(x,M)) = F \wedge open\_flag(M) = T$
  then $ex(type(x,M))$; Plan_x ; $ex(delete(x,M))$
  else $ex(undelete(x,M)))$; $ex(type(x,M))$; Plan_3 ; $ex(delete(x,M))$

Finally, the plan metavariable Plan_{new} of the current plan specification Spec_{new} is instantiated with the plan skeleton. In order to extend the plan skeleton to a correct plan, the instantiated plan specification is proven constructively. During the proof, metavariables are replaced by newly generated plans and action instantiations are verified if they indeed reach the desired goal.
The proof is guided by a special purpose tactic for second principles planning. If the tactic is unable to confirm an action instantiation, it calls another proof tactic which retracts instantiation decisions and flexibly modifies the plan skeleton. If the proof is successful, then a correct plan is obtained by modifying a plan from the plan library.
Apart from the refitting of a reused plan, the *optimization* of the plan, e.g. the deletion of superfluous actions from it, is supported in MRL.

4. Plan Modification in Action

The approach to deductive plan reuse has been implemented as part of the PHI system in SICSTUS PROLOG. In order to compare it with other approaches we used blocks world examples from the systems SPA[10] and PRIAR[13]. MRL and SPA were both running on the same Solbourne Sparc server, while the data for PRIAR are cited from [12].
The examples which we measured are simple: in the initial state all blocks are on the table and in the goal state all blocks are stacked on each other to form a tower. The blocks-world planning instances are named "$n$bs" where n is an integer parameter denoting the number of blocks which are involved.
We show as an example the solution of the 8bs planning problem by reuse of the 2bs, 3bs ... 8bs plans leading to the results in table 1:

Library	SPA			MRL			PRIAR
-	-	-	18533	-	-	820	79400
2bs	45	17317	17462	480	930	1410	-
3bs	78	13217	13295	760	870	1630	17400
4bs	128	10161	10289	1070	840	1910	15400
5bs	244	7361	7605	1490	810	2300	10100
6bs	567	5178	5745	1880	750	2630	8200
7bs	1645	2678	4321	2340	740	3080	6700
8bs	8117	111	8228	2610	0	2610	4200

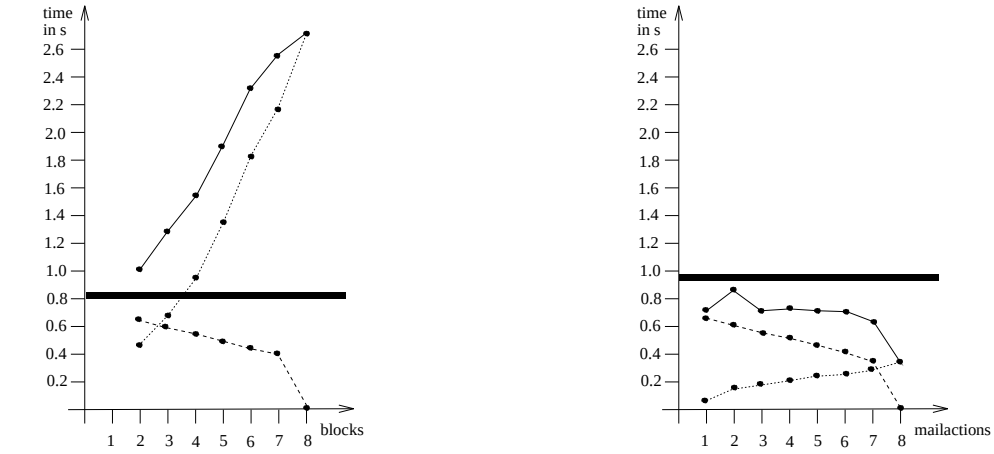
Table 1: Comparing MRL to PRIAR and SPA in the blocks world. For SPA and MRL we give three numbers. The first number is the time spent on plan interpretation, which is the proof attempt in MRL and a syntactic plan matching in SPA. The second number is the effort spent on plan refitting. The third number is the sum of both and represents the total effort on plan modification. For PRIAR we can only cite the total modification effort as reported in [12]. Each number gives the average CPU run times in milliseconds. The first row shows the time each of the planners spent for the generation of the 8bs plan from scratch.

Compared with the systems SPA and PRIAR, MRL shows a better performance. We see the following reasons for this speedup:

- The implementation of MRL is based on a formal theoretical model with clearly defined semantics.
- Logical proofs are guided by tactics in order to avoid unnecessary search effort.
- An appropriate instantiation of the reused plan is computed using an order-sorted unification algorithm.

On the other hand, solving the 8bs problem by plan reuse in MRL is more costly when compared with the PHI planner. This is caused by the efficiency of the PHI planner as well as by properties of the blocks world as figure 2 shows.
To highlight to influence of the application domain, examples from the mail domain are compared with blocks-world examples. It should be noted that we used the same proof tactics and order-sorted unification algorithm for the mail domain as well as for the blocks world.

The experiments demonstrate that the effort for planning from scratch and for plan refitting is almost the same for both problems, but they differ significantly in the effort which has to be spent on plan interpretation. In the blocks world, the proof attempt is much more expensive because the goal state description is very homogeneous, i.e. all objects are of the same sort. This leads to many different instantiation possibilities. In the mail domain we have fewer objects and they are of different sorts, which makes the proof attempt less expensive since the unification algorithm can benefit from the sort information.



a) MRL: blocks world

b) MRL: mail domain

Figure 2: Influence of the application domain in MRL. The horizontal bar indicates plan generation time, the dashed line plots the time for plan refitting, and the dotted line plots the time for plan interpretation. The solid line plots the resulting time for plan modification.

Impressive savings are achieved when the reused plan already provides a solution and no effort has to be spent on plan refitting. In such a situation, the proof performed during plan interpretation is *sufficient* for the immediate reuse of the plan and *no* additional verification of it as in SPA or PRIAR must be carried out.
We illustrate these savings in table 2 with some examples from the reuse of conditional plans in the UNIX mail domain where we compare
- the CPU time and the number of deduction rule applications which are necessary to generate the plan from scratch with
- the CPU time and the number of deduction rules which are necessary to prove that the reused plan is already a solution during plan interpretation.
The CPU run time savings are in general better than the savings of deduction rules because the rules applied during plan interpretation are in general less expensive than the generation rules the planner applies.
The empirical results can be summarized as follows:

- Deductive approaches to plan reuse lead to a very efficient behaviour when compared with non-deductive approaches.

- Efficiency gains in plan reuse depend on properties of the application domain as well as on the effort that has to be spent on plan refitting.

number of cases	length of subplan	number of plans	Plan Generation cpu	Plan Generation rules	Plan Reuse cpu	Plan Reuse rules	savings of rules	savings of cpu time
2	2	1	570	52	250	41	21.2 %	56.2 %
2	4	2	1290	120	330	59	50.9 %	74.5 %
2	5	2	1430	143	330	59	58.8 %	76.8 %
2	6	1	1700	152	610	86	43.5 %	64.2 %
3	3	1	2160	209	570	97	53.6 %	73.7 %
3	4	1	2620	257	590	97	62.3 %	77.5 %
3	10	16	9580	762	2310	229	70.0 %	75.9 %

Table 2: Savings with deductive plan interpretation. The first three columns in the table characterize the conditional plans, which were measured by the number of cases which are distinguished in each plan, the average length of the subplan for each case and the number of different plan hypotheses which were generated as possible solutions.

The results give some evidence that plan reuse is in particular useful in heterogeneous domains where stored standard solutions can be reused for frequently occurring complex planning problems (see also [17, 18]).

5. Conclusion

We have presented a new approach which extends deductive planners with the ability to reuse and modify plans. The approach comprises a completely logic-based formalization of plan reuse covering the modification, retrieval and representation of plans. It is independent from the underlying logic the deductive planning system relies on.
The formalization provides a reuse system with a clearly defined semantics and ensures that modified plans are provably correct. Apart from sequential plans, the system is able to reuse plans containing control structures like conditionals and iterations.
The practical experiences are encouraging: in comparison to experiences with other systems, deductive plan reuse has given very satisfactory results.

Acknowledgements

I would like to thank the members of the PHI research group, Mathias Bauer, Susanne Biundo, Dietmar Dengler and Gaby Paul for their interest in my work and for fruitful discussions.

References

[1] M. Bauer, S. Biundo, D. Dengler, J.Koehler, and G. Paul. PHI - a logic-based tool for intelligent help systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 460–466, Chambery,France, 1993. Morgan Kaufmann, Menlo Park.

[2] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[3] S. Biundo and D. Dengler. The logical language for planning LLP. Research Report, German Research Center for Artificial Intelligence, 1993.

[4] S. Biundo, D. Dengler, and J. Koehler. Deductive planning and plan reuse in a command language environment. In B. Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 628–632, Vienna, Austria, August 1992. John Wiley & Sons.

[5] *Proceedings of the 2nd Case-Based Reasoning Workshop*, Pensacola Beach, Florida, 1989. Morgan Kaufman, San Mateo.

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

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

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

[9] 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.

[10] S. Hanks and D. S. Weld. Systematic adaptation for case-based planning. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, pages 96–105, Washington, D.C., 1992.

[11] 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.

[12] 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.

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

[14] J. Koehler. An application of terminological logics to case-based reasoning. Research Report, German Research Center for Artificial Intelligence, 1993.

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

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

[17] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1436–1441, Chambery, France, August 1993. Morgan Kaufmann.

[18] 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.

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

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

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