Correct Modification of Complex Plans [1]

Jana Koehler[2]

Abstract. We present a general approach to flexible plan modification based on a deductive framework that enables a planner to correctly modify complex plans containing control structures like conditionals and iterations.

1 Introduction

Work on plan reuse and modification is motivated by expected gains in flexibility and efficiency when planners are provided with the ability to modify and reuse plans. The need to modify existing plans occurs when execution fails, where a plan has to be revised in the light of new information, and when a plan specification is changed, i.e., a plan has to be adapted to new requirements.

While there exist frameworks extending STRIPS-based planners, e.g., the PRIAR system [10] and the SPA system [7], there are no approaches which study plan reuse and modification in the context of *deductive planning* [6, 14, 2, 15].

Following a logical approach, plan modification 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 general formalism with clearly defined semantics is obtained and the formal view facilitates the investigation of theoretical properties [16, 17]. While current approaches are limited to the modification of sequential plans containing no control structures, this approach enables a system to automatically modify *sequential*, *conditional*, and *iterative* plans.

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 is done by a constructive proof of a specification formula Spec_{old} leading to a plan Plan_{old} that is *sufficient* for the specification. A plan Plan_{old} is a solution for another specification Spec_{new} if Spec_{old} entails Spec_{new}, i.e., each solution for Spec_{old} is a solution for Spec_{new}:

$$\mathsf{Spec}_{old} \models \mathsf{Spec}_{new}$$

If this relationship holds, then Spec_{new} is a logical instance of Spec_{old} and thus, solving Spec_{old} is sufficient for solving Spec_{new}. An instance of the reused plan Plan_{old} will therefore also solve the current planning problem.

A deductive proof attempt is suitable to show this relationship between the two plan specifications: in general, plan specifications comprise a description of the preconditions, *pre*, which hold in the initial state and a description of the goals, *goal*, the plan has to achieve. Therefore, we show $\mathsf{Spec}_{old} \models \mathsf{Spec}_{new}$ by attempting to prove that

$$pre_{new} \rightarrow pre_{old} \quad \text{and} \quad goal_{old} \rightarrow goal_{new}$$

hold. If both subproofs succeed we know that firstly, the current preconditions pre_{new} are sufficient for the preconditions pre_{old} the reused plan Plan_{old} requires and that secondly, the goals $goal_{old}$ achieved by Plan_{old} are sufficient for the currently required goals $goal_{new}$. This means the reused plan is applicable in the current initial state and it achieves at least all of the currently required goals.

The proof attempt enables a planner to compare planning problems deductively in order to find out whether the reuse candidate is "sufficiently similar" to the current planning problem, i.e., whether their preconditions and goals are to a large extent common. The comparison requires us to perform unification operations during the proof, i.e., we apply substitutions such that variables from Spec_{old} are instantiated with terms from Spec_{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.

Based on this formal framework, plan modification proceeds in two phases:

1. **Plan Interpretation:** The relation between the preconditions (precondition proof) and goals (goal proof) is proved. During the proof, knowledge about regularities of the application domain that is represented in the form of deduction rules can be applied. If the proof succeeds, an instance of Plan_{old} is computed satisfying Spec_{new}, otherwise refitting information is extracted from the failed proof.

2. **Plan Refitting:** Plan refitting tries to *modify* the reuse candidate in accordance with the result of the interpretation phase, i.e., it constructs a *plan skeleton* that the planner can exploit in generating a solution. The plan skeleton is extended to a correct plan by *verifying* reused subplans and by *generating* new subplans for open subgoals. Thus, plan refitting *constrains* the current planning process by providing the planner with reusable "pieces of information" in the form of action instances and control structures.

[1] This paper has been awarded the Digital Equipment Prize 1994.
[2] German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, e-mail: koehler@dfki.uni-sb.de

© 1994 J. Koehler
ECAI 94. *11th European Conference on Artificial Intelligence* Edited by A. Cohn
Published in 1994 by John Wiley & Sons, Ltd.

3 Deductive Plan Modification in MRL

The formal framework presented in the previous section provides the basis for the reuse component MRL [13, 11] that has been developed as an integrated part of the deductive planner PHI [4, 1]. The underlying planning formalism is the modal temporal logic LLP that possesses an interval-based semantics [3].

LLP provides the modal operators $\circ$ (next), $\Diamond$ (sometimes), $\Box$ (always) and the binary modal operator ; (chop) which expresses the sequential composition of formulae. The only "fluents" available in LLP are *local variables*, i.e., changes of states are reflected in changed values of these variables. *Plans* are represented by a certain class of LLP formulae. They may contain basic actions which are expressed by the *execute* predicate *ex* and control structures like *conditionals* (**if-then-else**) and iteration (**while**).

Plan generation is done deductively by performing constructive proofs of plan specifications in a sequent calculus which was developed for LLP. *Plan specifications* are LLP formulae of the form [Plan $\wedge$ *precondition*] $\rightarrow$ *goal*, i.e., if the Plan is carried out in a state where the *precondition* holds then a state will be achieved satisfying the *goal*. During the proof, the planvariable Plan is replaced by a plan formula satisfying the specification. The proofs are guided by tactics which can be described in a tactic language [3] provided by the system, an idea which was borrowed from the field of *tactical theorem proving* [5, 8, 18].

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. For example, the axiomatization of the *save*-command reads

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

The state of a mailbox may change from state to state, i.e., mailboxes are represented as local variables. As a precondition, the *save*-command requires that the mailbox m 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 message x from the mailbox m is saved in a file f and its *save_flag* is set to the value *true* in the next state.

In the following, we discuss an example in which an iterative plan is modified. Assume that the current planning problem Spec_{new} is specified as

$$\begin{aligned} & \mathsf{Plan} \wedge open\_flag(mb) = T \wedge \\ & \quad \forall\, mail\,[\,sender(msg(mail,mb)) = Joe \\ & \qquad \rightarrow delete\_flag(msg(mail,mb)) = F\,] \\ & \rightarrow \Diamond \forall\, mail\,[\,sender(msg(mail,mb)) = Joe \\ & \qquad\quad \rightarrow read\_flag(msg(mail,mb)) = T \wedge \\ & \qquad\qquad save\_flag(msg(mail,mb)) = T \wedge \\ & \qquad\qquad delete\_flag(msg(mail,mb)) = T\,] \end{aligned} \tag{1}$$

The specification describes the goal "Read, save and delete all messages from sender Joe" under the precondition that the mailbox is open and no message from Joe has yet been deleted. Assume that furthermore, the search in the plan library as described in [11] has retrieved the reuse candidate Spec_{old}

$$\begin{aligned} & \mathsf{Plan} \wedge open\_flag(m) = T \wedge \\ & \quad \forall\, x\,[\,sender(msg(x,m)) = s \\ & \qquad \rightarrow delete\_flag(msg(x,m)) = F\,] \\ & \rightarrow \Diamond[screen\_display = select\_msgs(s,m) \wedge \\ & \qquad \Diamond \forall\, x\,[\,sender(msg(x,m)) = s \\ & \qquad\qquad \rightarrow read\_flag(msg(x,m)) = T \wedge \\ & \qquad\qquad\quad file(msg(x,m)) = f\,]] \end{aligned} \tag{2}$$

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

```
$ex(from(s,m))\,;n := 1\,;$
while $n < length(m)$ do
    if $sender(msg(n,m)) = s$
      then $ex(type(n,m))\,;ex(save(n,m))$
      else $ex(empty\_action)\,;$
$n := n + 1$ od
```

In the example, an iterative plan has to be modified, in which a case analysis occurs. The plan specifications contain universally quantified conjunctive goals, disjunctive and negated atomic preconditions and goals as well as the specification of temporary subgoal states with the help of nested *sometimes* operators.[3]

3.1 Plan Interpretation

The plan interpretation phase 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 in most plan reuse systems. Performing a proof of the formulae can be viewed as a semantic comparison. During a 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 we can derive the rules

$$\begin{aligned} & save\_flag(msg(x,m)) = T \\ & \qquad \rightarrow file(msg(x,m)) = f \end{aligned} \tag{3}$$

$$\begin{aligned} & file(msg(x,m)) = f \rightarrow \\ & \qquad save\_flag(msg(x,m)) = T \end{aligned} \tag{4}$$

reflecting the relationship between the atomic effects of this command, i.e., 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.

The validity of the relation between the preconditions pre_{new} and pre_{old} is obvious. The proof of the relation between the goals requires the following sequent in the LLP sequent calculus to be proved:

[3] Observe that $A \rightarrow B$ is equivalent to $\neg A \vee B$, i.e., a goal containing an implication can be considered as a disjunctive goal.

$$\begin{aligned}&\Diamond[\mathit{screen\_display} = \mathit{select\_msgs}(s,m) \wedge\\ &\quad \Diamond \forall x\,[\,\mathit{sender}(\mathit{msg}(x,m)) = s\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(x,m)) = T \wedge\\ &\qquad\quad \mathit{file}(\mathit{msg}(x,m)) = f\,]\,]\\ &\Rightarrow\\ &\Diamond \forall \mathit{mail}\,[\,\mathit{sender}(\mathit{msg}(\mathit{mail},mb)) = \mathit{Joe}\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{save\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{delete\_flag}(\mathit{msg}(\mathit{mail},mb)) = T\,]\end{aligned} \tag{5}$$

A special-purpose proof tactic guides the proof attempt in the LLP sequent calculus. The use of tactics supports the declarative representation of control knowledge. 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, i.e., we give up completeness in order to achieve efficiency [16]. Figure 1 sketches a part of the tactic *goal_tac* that is used in the example.

```
goal_tac(goal_seq,Axioms):-
  apply_rule_strict(left_sometimes,goal_seq,seq1),
  iterate_rule(left_and,seq1,seq2),
  or_else(call_tac(close_leaves,seq2,Axioms),
          call_tac(goal_tac,seq2,Axioms)).

close_leaves(seq2,Axioms):-
  apply_rule_strict(right_sometimes,seq2,seq3),
  call_tac(derive_axioms,seq3,Axioms).
```

Figure 1. Sketch of the goal-proof Tactic

The tactic specifies a well defined ordering of deduction rule applications. It is composed of *tacticals* [3] like *iterate_rule*, *apply_rule_strict*, and *call_tac*. Each tactical specifies a specific mode of rule or tactic applications. 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. The tactical *iterate_rule* repeats a rule application as long as possible, while the tactical *call_tac* calls another tactic. The following sequent rules are applied by the tactic *goal_tac*:

- $\dfrac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta}\ \mathit{left\_and}\ (l\wedge)$
- $\dfrac{\Gamma \Rightarrow A, \Delta \qquad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta}\ \mathit{right\_and}\ (r\wedge)$
- $\dfrac{\Gamma^*, A \Rightarrow \Delta^*}{\Gamma, \Diamond A \Rightarrow \Delta}\ \mathit{left\_sometimes}\ (l\Diamond)$ [4]
- $\dfrac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow \Diamond A, \Delta}\ \mathit{right\_sometimes}\ (r\Diamond)$

The proof tactic is designed in such a way that it always terminates. In addition, it is considered as a decision procedure: if the tactic does not construct a proof tree, it is assumed that no proof is possible and that a falsifying valuation for some of the leaves in the deduction tree has been obtained.

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

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 $goal_{new}$.

In the example, the tactic is called with sequent 5 as input argument. In a first step, it has to apply rule $l\Diamond$ and rule $l\wedge$ to sequent 5 which lead to sequent 6:

$$\begin{aligned}&\boxed{\mathit{screen\_display} = \mathit{select\_msgs(s,m)}},\\ &\Diamond \forall x\,[\,\mathit{sender}(\mathit{msg}(x,m)) = s\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(x,m)) = T \wedge\\ &\qquad\quad \mathit{file}(\mathit{msg}(x,m)) = f\,]\\ &\Rightarrow\\ &\Diamond \forall \mathit{mail}\,[\,\mathit{sender}(\mathit{msg}(\mathit{mail},mb)) = \mathit{Joe}\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{save\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{delete\_flag}(\mathit{msg}(\mathit{mail},mb)) = T\,]\end{aligned} \tag{6}$$

With the framed formula, the first temporary subgoal from $goal_{old}$ has been isolated. In the example, the specification of $goal_{new}$ contains only one temporary subgoal. The goal tactic calls now the tactic *close_leaves* that tries to prove a subgoal from $goal_{new}$ in using the isolated subgoal from $goal_{old}$. Obviously, this is impossible and such the tactic terminates with a failure. As *close_leaves* failed, the goal tactic is recursively called with sequent 6 as input argument. Applying the rule $l\Diamond$ to sequent 6 again leads to sequent 7. Note that rule $l\wedge$ cannot be applied and thus leaves the sequent unchanged.

$$\begin{aligned}&\forall x\,[\mathit{sender}(\mathit{msg}(x,m)) = s\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(x,m)) = T \wedge\\ &\qquad\quad \mathit{file}(\mathit{msg}(x,m)) = f\,]\\ &\Rightarrow\\ &\Diamond \forall \mathit{mail}\,[\,\mathit{sender}(\mathit{msg}(\mathit{mail},mb)) = \mathit{Joe}\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{save\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{delete\_flag}(\mathit{msg}(\mathit{mail},mb)) = T\,]\end{aligned} \tag{7}$$

Now, the tactic *close_leaves* applies the rule $r\Diamond$ leading to sequent 8:

$$\begin{aligned}&\forall x\,[\mathit{sender}(\mathit{msg}(x,m)) = s\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(x,m)) = T \wedge\\ &\qquad\quad \mathit{file}(\mathit{msg}(x,m)) = f\,]\\ &\Rightarrow\\ &\forall \mathit{mail}\,[\,\mathit{sender}(\mathit{msg}(\mathit{mail},mb)) = \mathit{Joe}\\ &\qquad \rightarrow \mathit{read\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{save\_flag}(\mathit{msg}(\mathit{mail},mb)) = T \wedge\\ &\qquad\quad \mathit{delete\_flag}(\mathit{msg}(\mathit{mail},mb)) = T\,]\end{aligned} \tag{8}$$

The tactic *derive_axioms* calls a specialized tactic dealing with universally quantified goals. This tactic aims at isolating and comparing each of the atomic subgoals occurring in $goal_{old}$ and $goal_{new}$. Consequently, the atomic subformulae must be separated by further sequent rule applications. The tactic applies the rules $r\,\forall$, $l\,\forall$ followed by the rules $r \rightarrow$ and $l \rightarrow$ as shown below.

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

- $$\frac{\Gamma \Rightarrow \Delta, A[a/x]}{\Gamma \Rightarrow \Delta, \forall x A}\; r\forall$$
- $$\frac{\Gamma \Rightarrow A, \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \rightarrow B \Rightarrow \Delta}\; l\rightarrow$$
- $$\frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow A \rightarrow B, \Delta}\; r\rightarrow$$

```
universal_goal_tac(subgoal,Axioms):-
    apply_rule_strict(r∀,subgoal,seq1),
    apply_rule_strict(l∀,seq1,seq2)
    apply_rule_strict(r →,seq2,seq3)
    apply_rule_strict(l →,seq3,seq4)
    iterate_rule(l∧,seq4,seq5),
    iterate_rule(r∧,seq5,outseq),
    call_tac(find_axioms,outseq,Axioms).
```

Figure 2. Sketch of the universal-goal Tactic

The tactic as shown in Figure 2 computes the sequents (A1) to (A3). They lead to axioms under the substitution $\{x/mail, m/mb, s/Joe\}$. In order to obtain an axiom from (A3), the knowledge represented in rule 4 is additionally applied.

(A1) $sender(msg(mail, mb)) = Joe$
$\Rightarrow sender(msg(x, m)) = s$

(A2) $read\_flag(msg(x, m)) = T$
$\Rightarrow read\_flag(msg(mail, mb)) = T$

(A3) $file(msg(x, m)) = f$
$\Rightarrow save\_flag(msg(mail, mb)) = T$

Observe that there is no axiom containing the current atomic subgoal $delete\_flag(msg(mail, mb)) = T$. Therefore, 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

Plan refitting starts with an analysis of the plan interpretation phase. The proof of the relation between the old and current preconditions $pre_{new} \rightarrow pre_{old}$ has been successfully completed, i.e., the reuse candidate is applicable in the current initial state. In order to analyze the goal proof, *entries* are computed which record the result of the proof attempt by relating atomic goals from $goal_{old}$ and from $goal_{new}$ to each other, cf. Figure 3. This information is now analyzed under the following assumptions:

- An atomic subgoal from $goal_{old}$ that occurs <u>not</u> in an axiom indicates that the reuse candidate may contain superfluous actions. The subgoal was not necessary to derive axioms, i.e., plan refitting concludes that this subgoal is not required in the current goal specification $goal_{new}$. Therefore, actions achieving this old subgoal are removed from the candidate plan.
- An atomic subgoal from $goal_{old}$ that occurs in an axiom indicates reusable actions. All actions achieving these subgoals are preserved because they achieve subgoals required in the current goal specification.
- An atomic subgoal from $goal_{new}$ that occurs <u>not</u> in an axiom indicates an open current subgoal not achieved by the reuse candidate. Plan refitting marks a failure of plan interpretation and starts with constructing a plan skeleton.

old new	$\langle screen\_display = select\_msgs(s, m),$ $-\rangle$	no Axiom
old new	$\langle read\_flag(msg(x, m)) = T,$ $read\_flag(msg(mail, mb)) = T\rangle$	Axiom
old new	$\langle file(msg(x, m)) = f,$ $save\_flag(msg(mail, mb)) = T\rangle$	Axiom
old new	$\langle -,$ $delete\_flag(msg(mail, mb)) = T\rangle$	no Axiom

Figure 3. Result of the Plan Interpretation Phase

Figure 4 shows the algorithm that analyzes the information represented in Figure 3.

```
for $i = 1$ to $n$ do /* all goals $goal_{new_i}$ */
    if entry $\langle goal_{new_i}, goal_{old_j}\rangle$ exists
        then reuse subplan achieving $goal_{old_j}$
        /* $goal_{new_i}$ is achieved by reuse candidate */
        else plan has to be modified
        /* generate new subplan for $goal_{new_i}$ */
endfor

for $j = 1$ to $k$ do /* all goals $goal_{old_j}$ */
    if entry $\langle goal_{new_i}, goal_{old_j}\rangle$ exists
        then reuse subplan achieving $goal_{old_j}$
        else attempt to optimize plan
        /* remove superfluous subplan for $goal_{old_j}$ */
endfor
```

Figure 4. Analysis of the Plan Interpretation Phase

Note that we have no information obtained from the failed proof attempt that is *sufficient* for constructing the "right" plan skeleton, i.e., a plan skeleton that can be extended to a correct plan. Therefore, plan refitting and plan generation cannot be separated and plan generation becomes involved in the process of plan-skeleton construction.

Plan refitting also relies on information from the plan library that is stored together with the reuse candidate: an analysis of the plan generation process that has led to the reused plan reveals which action from Plan_{old} achieves which atomic effect in $goal_{old}$. In the example, the following relations between actions and atomic goals are stored in the plan library:

- $ex(type(n, m))$ achieves $read\_flag(msg(x, m)) = T$

- $ex(save(n,m))$ achieves $file(msg(x,m)) = f$
- $ex(from(s,m))$ achieves
  $screen\_display = select\_msgs(s,m)$

Consequently, the action $ex(from(s,m))$ is removed from the skeleton, while $ex(type(n,m))$ and $ex(save(n,m))$ are preserved after proper instantiation with the substitution computed during the proof attempt. Furthermore, a plan for the missing subgoal $delete\_flag(msg(mail,mb)) = T$ has to be generated and inserted into the plan skeleton. In order to determine the position in the skeleton where this plan has to be added, the current goal specification $goal_{new}$ is analyzed with the help of the underlying planning system, as described in [13]. Finally, control structures must be verified and eventually modified during the refitting process. A special-purpose tactic for second-principles planning implements *verification* and *generation* as deductive processes. In the example, the tactic identifies the universally quantified goal in a first step and thus calls the tactic for the generation of an iterative plan by introducing a *while* construct which coincides with the iterative control structure in the skeleton. The proof that has led to this *while* construct is replayed. During the replay process the tactic has to construct the conditional plan occurring in the body of the iteration by introducing a case analysis. This process requires us to address the conjunctive goal containing the three atomic subgoals. After ordering the subgoals with the help of a tactic dealing with conjunctive goals [3], a plan for each of the subgoals is generated. This leads to a reuse of the subplan $ex(type(n,mb))\,;ex(save(n,mb))$ from the skeleton, while the action $ex(delete(n,mb))$ has to be generated from scratch.

As a result, the following plan solving Spec_{new} is obtained:

```
$n := 1\,;$
while $n < length(mb)$ do
        if $sender(msg(n,mb)) = Joe$
           then $ex(type(n,mb))\,;$
                $ex(save(n,mb))\,;$
                $\boxed{ex(delete(n,mb))}$
           else $ex(empty\_action)\,;$
$n := n + 1$ od
```

4 Conclusion

We have presented a deductive approach to plan modification which yields provably correct plans. Apart from sequential plans, this approach enables a planner to modify plans containing control structures like conditionals and iterations. The theoretical model is independent of any particular planning formalism and makes no restrictive assumptions on the nature of plans.

The approach is the basis for the implemented system MRL. As a main advantage, the system possesses a clearly defined semantics which allows formal properties of plan modification to be proved. Furthermore, an empirical evaluation of the system has led to very satisfactory results [13, 12].

ACKNOWLEDGEMENTS

I would like to thank Mathias Bauer, Susanne Biundo, Dietmar Dengler, Gaby Paul, and Wolfgang Wahlster for their interest in my work and for fruitful discussions. The anonymous referees made helpful comments on a draft version of this paper.

REFERENCES

[1] M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul, 'PHI - a logic-based tool for intelligent help systems', In IJCAI-93 [9], pp. 460–466.

[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, (1994).

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

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

[6] C. Green, 'Application of theorem proving to problem solving', in *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pp. 219–239, Washington, D.C., (May 1969).

[7] S. Hanks and D. Weld, 'Systematic adaptation for case-based planning', in *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, pp. 96–105, Washington, D.C., (1992). Morgan Kaufmann, San Mateo.

[8] 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, pp. 117–131, Kaiserslautern, Germany, (1990). Springer, Berlin.

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

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

[11] J. Koehler, 'An application of terminological logics to case-based reasoning', in *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, eds., J. Doyle, E. Sandewall, and P. Torasso. Morgan Kaufmann, San Francisco, CA, (1994).

[12] 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.

[13] J. Koehler, 'Flexible plan reuse in a formal framework', in *Current Trends in AI Planning*, eds., C. Bäckström and E. Sandewall, pp. 171–184. IOS Press, Amsterdam, Washington, Tokyo, (1994).

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

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

[16] B. Nebel and J. Koehler, 'Plan modification versus plan generation: A complexity-theoretic perspective', In IJCAI-93 [9], pp. 1436–1441.

[17] 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).

[18] L. Paulson, 'Isabelle: The next 700 theorem provers', in *Logic and Computer Science*, ed., P. Odifredi, Academic Press, (1990).