Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-48

Plan Modification versus Plan Generation: A Complexity-Theoretic Perspective

Bernhard Nebel, Jana Koehler

October 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, 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, Philips, SEMA Group Systems, Siemens and Siemens-Nixdorf. 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
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

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.

Prof. Dr. Gerhard Barth
Director

Plan Modification versus Plan Generation: A Complexity-Theoretic Perspective

Bernhard Nebel, Jana Koehler

DFKI-RR-92-48

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8901 8 and ITW 9000 8).

Plan Modification versus Plan Generation: A Complexity-Theoretic Perspective*

Bernhard Nebel Jana Koehler
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11, Germany
phone: +49 (681) 302-5254/-5259
e-mail: {nebel|koehler}@dfki.uni-sb.de

October 19, 1992

Abstract

The ability of a planner to modify a plan is considered as a valuable tool for improving efficiency of planning by avoiding the repetition of the same planning effort. From a computational complexity point of view, however, it is by no means obvious that modifying a plan is computationally as easy as planning from scratch if the modification has to follow the principle of "conservatism," i.e., to reuse as much of the old plan as possible. Indeed, considering propositional STRIPS planning, it turns out that conservative plan modification is as hard as planning and can sometimes be harder than plan generation. Furthermore, this holds even if we consider modification problems where the old and the new goal specification are similar. We put these results into perspective and discuss the relationship to existing plan modification systems. Although sometimes claimed otherwise, these systems do not address the modification problem, but use a non-conservative form of plan modification as a heuristic technique.

*This work was supported by the German Ministry for Research and Technology (BMFT) under contracts ITW 8901 8 and ITW 9000 8 as part of the WIP project and the PHI project.

Contents

1 Introduction

Plan generation in complex domains is normally a resource and time consuming process. One way to improve the efficiency of planning systems is to avoid the repetition of planning effort whenever possible. For instance, in situations when the goal specification is changed during plan execution or when execution time failures happen, it seems more reasonable to *modify* the existing plan than to plan from scratch again. In the extreme, one might go as far as basing the entire planning process on plan modification, a method that could be called *planning from second principles.*

Instead of generating a plan from scratch, that method tries to exploit knowledge stored in previously generated plans. The current problem instance is used to find a plan in a plan library that—perhaps after some modifications—can be used to solve the problem instance at hand. Current approaches try to integrate methods from analogical or case-based reasoning to achieve a higher efficiency [Hammond, 1990; Veloso, 1992], integrate domain-dependent heuristics [Howe, 1992] or investigate reuse in the general context of deductive planning [Koehler, 1992; Biundo *et al.*, 1992].

Some experiments give evidence that planning based on second principles might indeed be more efficient than planning from scratch [Kambhampati and Hendler, 1992; Veloso, 1992; Hanks and Weld, 1992]. However, it is by no means clear in how far these results generalize. In fact, it is not obvious that *modifying* an existing plan is computationally as easy as *generating* one from scratch, in particular, if we adopt the principle of *conservatism* [Kambhampati and Hendler, 1992], that is to try to recycle "as much of the old solution as possible" [Veloso, 1992, p. 133] or to "produce a plan ... by minimally modifying [the original plan]" [Kambhampati and Hendler, 1992, p. 196].

Considering, for instance, the revision of logical theories, most revision schemata turn out to be computationally harder than deduction [Nebel, 1991; Eiter and Gottlob, 1992].[1] A similar result holds for abduction [Selman and Levesque, 1990], which may be viewed as "modifying the assumptions in a proof." Hence, it seems worthwhile to have a closer look at the computational nature of the process of modifying a plan in order to find out why and under which circumstances plan modification and reuse promises to be more efficient than planning from scratch.

The computational complexity of different forms of planning has been recently analyzed by a number of authors [Chapman, 1987; Bäckström and Klein, 1991; Bylander, 1991; Chenoweth, 1991; Gupta and Nau, 1991; By-

[1] More precisely, revision is in most cases Π_2^p-complete. Assuming, as is customary, that the polynomial hierarchy does not collapse (see, e.g., [Garey and Johnson, 1979; Johnson, 1990]), this implies that revising a propositional theory is harder than doing deduction, which is Π_1^p- or co-NP-complete.

lander, 1992a; Erol *et al.*, 1992]. However, the computational complexity of plan modification has not been investigated yet. We will analyze this problem in the formal framework of *propositional STRIPS planning* as defined by Bylander [1991; 1992a]. As Bylander [1991] notes, this model of planning is "impoverished compared to working planners" and is only intended to be a "tool for theoretical analysis." However, since we are mainly interested in *comparing* plan generation with plan modification from a *complexity-theoretic perspective*, this framework is appropriate for our purposes.

As it turns out, modifying a plan is not easier than planning from scratch. On the positive side, we show that modification does not add any complexity to planning if we consider the *general case*. However, there exist special cases when modifying a plan *conservatively*, i.e., *by using as much of the old plan as possible*, can be harder than creating one from scratch, as we will show. This means that plan modification *is not uniformly as easy as plan generation*. Further, we show that these results also hold if we assume that the old and the new planning situation are similar.

Putting these results into perspective and relating them to practical approaches reveals that these approaches do not address the plan modification problem at all, although some authors claim otherwise.

The paper is organized as follows. In Section 2, we define the notion of propositional STRIPS planning following Bylander [1991] and recapitulate the main results. Based on that, we introduce a formal model of plan modification in Section 3. In Section 4 we analyze the computational complexity of different modification problems relative to their corresponding planning problems. The problem of plan modification restricted to the case where the old planning situation is similar the new one is analyzed in Section 5. Finally, in Section 6 we discuss the relationships between our results and practical experiences in plan modification and reuse.

2 Propositional STRIPS Planning

Like Bylander [1991], we define an instance of propositional planning as a tuple $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where:

- $\mathcal{P}$ is a finite set of ground atomic formulae, the *conditions*,
- $\mathcal{O}$ is a finite set of *operators*, where each operator $o \in \mathcal{O}$ has the form $o^+, o^- \Rightarrow o_+, o_-$, where
  - $o^+ \subseteq \mathcal{P}$ are the *positive preconditions*,
  - $o^- \subseteq \mathcal{P}$ are the *negative preconditions*,
  - $o_+ \subseteq \mathcal{P}$ are the *positive postconditions* (add list), and
  - $o_- \subseteq \mathcal{P}$ are the *negative postconditions* (delete list).

- $\mathcal{I} \subseteq \mathcal{P}$ is the *initial state*, and
- $\mathcal{G} = \langle \mathcal{G}_+, \mathcal{G}_- \rangle$ is the *goal specification* with $\mathcal{G}_+ \subseteq \mathcal{P}$ the positive goals and $\mathcal{G}_- \subseteq \mathcal{P}$ the negative goals.

$\mathcal{P}$ is the set of relevant conditions. A *state* is a subset $S \subseteq \mathcal{P}$ with the intended meaning that $p \in \mathcal{P}$ is true in state S if $p \in S$, false otherwise. $\mathcal{O}$ is the set of *operators* that can change states. $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal state specification, with the intended meaning that all conditions $p \in \mathcal{G}_+$ must be true and all conditions $p \in \mathcal{G}_-$ must be false. A plan Δ is a finite sequence $\langle o_1, \ldots, o_n \rangle$ of *plan steps* $o_i \in \mathcal{O}$. An operator may occur more than once in a plan. A plan Δ *solves* an instance Π of the planning problem iff the *result* of the application of Δ to $\mathcal{I}$ leads to a state S that satisfies the goal specification $\mathcal{G}$, where the result of applying $\Delta = \langle o_1, \ldots, o_n \rangle$ to a state S is defined by the following function:

$$
\begin{aligned}
\mathit{Result}: (2^{\mathcal{P}} \cup \bot) \times \mathcal{O}^* &\rightarrow 2^{\mathcal{P}} \cup \bot \\
\mathit{Result}(S, \langle\rangle) &= S \\
\mathit{Result}(S, \langle o \rangle) &= \begin{cases} (S \cup o_+) - o_- & \text{if } o^+ \subseteq S \wedge o^- \cap S = \emptyset \\ \bot & \text{otherwise} \end{cases} \\
\mathit{Result}(S, \langle o_1, o_2, \ldots, o_n \rangle) &= \mathit{Result}(\mathit{Result}(S, \langle o_1 \rangle), \langle o_2, \ldots, o_n \rangle)
\end{aligned}
$$

In other words, if the precondition of an operator is satisfied by a state, the positive postconditions are added and the negative postconditions are deleted. Otherwise, the state becomes *undefined*, denoted by $\bot$.[2]

As usual, we consider *decision problems* in order to analyze the computational complexity of planning.[3] PLANSAT is defined to be the *decision problem* of determining whether an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ of propositional STRIPS planning has a solution, i.e., whether there exists a plan Δ such that $\mathit{Result}(\mathcal{I}, \Delta)$ satisfies the goal specification. PLANMIN [Bylander, 1992b] is defined to be the problem of determining whether there exists a solution of length n or less, i.e., it is the decision problem corresponding to the *search problem* of generating plans with minimal length.

Based on this framework, Bylander [1991; 1992a; 1992b] analyzed the computational complexity of the general propositional planning problem and a number of generalizations and restricted problems. In its most general form, both PLANSAT and PLANMIN are PSPACE-complete. Severe restrictions on the form of the operators are necessary to guarantee polynomial time or even NP-completeness.

[2] This is a slight deviation from Bylander's [1991] definition that does not affect the complexity of planning. This deviation is necessary, however, to allow for a meaningful definition of the plan modification problem.

[3] We assume that the reader is familiar with the basic notions of complexity theory as presented, for instance, in [Garey and Johnson, 1979].

3 Plan Modification in a Propositional Framework

Kambhampati and Hendler [1992] define the *plan modification problem* as follows (adapted to our framework of propositional STRIPS planning):

> Given an instance of the planning problem $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ and a plan Δ that solves the instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, produce a plan Δ' that solves Π' by *minimally modifying* Δ.

We will call this problem MODGEN.

By "minimal modification of a plan" Kambhampati and Hendler [1992] mean to "salvage as much of the old plan as possible." Other authors are less explicit about what they mean by modifying a plan, but the idea to use as much of the old plan as possible for solving the new problem instance seems to be customary in order to minimize the additional planning effort [Veloso, 1992, p. 133].

Another conceivable interpretation of "minimal modification," namely, of additionally adding as few plan steps as possible, is usually not considered. The reason for not imposing this constraint is obvious. This requirement would make modification as hard as finding an optimal plan, i.e., as hard as PLANMIN, because in this case PLANMIN reduces to modification for the limiting case of an empty modification candidate. Since most plan-reuse systems are only aimed at satisficing instead of optimal solutions, such a requirement would in fact run counter to the idea of reducing planning effort.

Turning the above specified search problem into a decision problem leads to what we will call the MODSAT problem:

> An instance of the MODSAT problem is given by $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$, a plan Δ that solves $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, and an integer $k \leq |\Delta|$. The question is whether there exists a plan Δ' that solves Π' and *contains a subplan of* Δ *of at least length* k?

In order to fully specify MODSAT, we have to define the meaning of the phrase "Δ' contains a subplan of Δ of length k." For this purpose, we define the notion of a *plan skeleton*, a sequence of operators and "wildcards," denoted by "$*$." The length of a plan skeleton is the number of operators, i.e., we ignore the wildcards. A plan skeleton can be *derived* from a plan according to a *modification strategy* $\mathcal{M}$ by deleting and rearranging plan steps and adding wildcards. A plan skeleton can be *extended* to a plan by replacing each wildcard by a possibly empty sequence of operators. Now we say that *plan* Δ' *contains a subplan of* Δ *of length* k *according to a modification strategy* $\mathcal{M}$ iff a skeleton Γ of length k can be derived from Δ according to $\mathcal{M}$ and Γ can be extended to Δ'. In general, we will consider only *polynomial-time*

modification strategies, i.e., strategies such that verifying that the skeleton Γ can be derived from the plan Δ is a polynomial-time problem. In the following, we will consider three different plan modification strategies that satisfy this constraint.

The first alternative we consider is to allow for deletions in the original plan and additions before and after the original plan. Supposing the plan

$$\Delta = \langle o_1, \ldots, o_i, o_{i+1}, \ldots, o_{j-1}, o_j, \ldots, o_n \rangle,$$

the following plan skeleton could be derived from Δ, for instance:

$$\Gamma = \langle *, o_1, \ldots, o_i, o_j, \ldots, o_n, * \rangle,$$

where Γ has length $i + n - j + 1$. The corresponding modification problem will be called MODDEL.

The second alternative is to allow for deletion of plan steps in the old plan and additions before, after, and in the middle of the old plan. Assuming the same plan Δ as above, the following skeleton plan of length $i + n - j + 1$ could be derived:

$$\Gamma = \langle *, o_1, \ldots, o_i, *, o_j, \ldots, o_n, * \rangle.$$

The corresponding modification problem is called MODDELINS.

The final alternative is to count the number of plan steps in the plan skeleton Γ that also appear in the old plan Δ without considering the order. In other words, we view Δ and Γ as multisets and take the cardinality of the intersection as the number of old plan steps that appear in the new plan. The corresponding modification problem is called MODMIX. Although this model of modification may seem to give away too much of the structure of the old plan, "changing step order" is considered to be a reasonable modification operation (see, e.g., [Hanks and Weld, 1992, p.96]).

Finally, it should be noted that although the framework we have defined above deals only with linear plans, it can be easily modified to apply to nonlinear planning, as well. In particular, all hardness results will apply directly to nonlinear planning since linear plans are simply a special case of nonlinear ones.

4 The Complexity of Plan Modification

One almost immediate consequence of the definitions above is that plan modification cannot be easier than plan generation. This even holds for all restrictions of the PLANSAT problem. If PLANSAT$_\rho$ is a restricted planning problem, then MODSAT$_\rho$ shall denote the corresponding modification problem with the same restrictions.

Proposition 1 PLANSAT$_\rho$ *transforms polynomially to* MODSAT$_\rho$ *for all restrictions* ρ.

Proof. The restriction of MODSAT$_\rho$ to empty old plans and $k = 0$ is identical to PLANSAT$_\rho$. ∎

However, plan modification is also not harder than plan generation in the general case.

Proposition 2 MODSAT *is* PSPACE*-complete.*

Proof. Because of Proposition 1 and the fact that PLANSAT is PSPACE-complete [Bylander, 1991, Theorem 1], MODSAT is PSPACE-hard.

MODSAT is in NPSPACE because (1) guessing a skeleton Γ of length k and verifying that it can be derived from the old plan Δ and (2) guessing step by step (with a maximum of $2^{|\mathcal{P}|}$ steps) a new plan Δ' and verifying that it solves the instance Π' and extends Γ can be obviously done in polynomial space. Since NPSPACE = PSPACE, it follows that MODSAT $\in$ PSPACE. ∎

This proposition could be taken as evidence that plan modification is not harder than plan generation. However, it should be noted that the proposition is only about the general problem. So, it may be the case that there exist special cases such that plan modification is harder than generation. Such a case will not be found among the PSPACE- and NP-complete planning problems, however.

Theorem 3 *If* PLANSAT$_\rho$ *is a restricted planning problem that is* PSPACE*-complete or* NP*-complete, then* MODSAT$_\rho$ *is* PSPACE*-complete or* NP*-complete problem, respectively.*[4]

Proof. PSPACE-hardness and NP-hardness, respectively, are obvious because of Proposition 1. Membership follows in case of PSPACE by Proposition 2. In case of NP, we initially guess (1) n ($0 \leq n \leq |\Delta| + 2$) possibly empty plans Δ_i such that $|\Delta_i| \leq |\Delta|$, (2) $2n$ states $S_1, \ldots, S_{2n}$, and (3) n polynomially bounded proofs that there exists plans from each state S_{2i} to state S_{2i+1} for $1 \leq i \leq n-1$. Since PLANSAT$_\rho$ is in NP, such proofs exist (in most cases, these proofs will be plans). Then we verify in polynomial time (1) that $S_1 = \mathcal{I}$ and S_{2n} satisfies the goal specification $\mathcal{G}$, (2) that $Result(S_{2i-1}, \Delta_i) = S_{2i}$, (3) that the plan existence proofs are correct, and (4) that $\langle \Delta_1, *, \Delta_2, *, \ldots, \Delta_{n-1}, *, \Delta_n \rangle$ is a skeleton of length k that can be derived from Δ. This is obviously a nondeterministic algorithm that runs in polynomial time. ∎

[4]Note that the proof also applies to Σ_n^p-complete planning problems. There are no such planning problems known yet, however.

The converse of the above theorem does not hold, however. There exist cases when plan generation is a polynomial time problem while plan modification is NP-complete.

Theorem 4 *There exists a polynomial-time* PLANSAT$_\rho$ *problem such that the corresponding* MODDEL$_\rho$ *and* MODDELINS$_\rho$ *problems are* NP*-complete.*

Proof. The planning problem PLANSAT$_1^+$ defined by restricting operators to have only positive preconditions and only one postcondition can be solved in polynomial time [Bylander, 1991, Theorem 7]. Let PLANSAT$_1^{+,\overline{post}}$ be the planning problem defined by restricting operators to have (1) only one postcondition p, (2) the negated condition $\overline{p}$ as a precondition, and (3) any number of additional positive preconditions. From the specification of the algorithm Bylander [1991] gives for PLANSAT$_1^+$, it is evident that PLANSAT$_1^{+,\overline{post}}$ can also be solved in polynomial time. We will show that the corresponding modification problems MODDEL$_1^{+,\overline{post}}$ and MODDELINS$_1^{+,\overline{post}}$ are NP-complete.

For the hardness part we use a reduction from SAT, the problem of satisfying a boolean formula in conjunctive normal form. Let $V = \{v_1, \ldots, v_m\}$ be the set of boolean variables and let $C = \{c_1, \ldots, c_n\}$ be the set of clauses. Now we construct a MODDEL$_1^{+,\overline{post}}$ problem that can be satisfied iff there exists a satisfying truth assignment for the SAT problem.

The set of conditions $\mathcal{P}$ contains the following ground atoms:

T_i,	$1 \leq i \leq m$,	v_i = true has been selected
F_i,	$1 \leq i \leq m$,	v_i = false has been selected
S_i,	$1 \leq i \leq m$,	the truth value for v_i has been selected
E_i,	$0 \leq i \leq m$,	enable evaluation
C_j,	$1 \leq n \leq n$,	c_j evaluates to true.

Further, we assume the following set of operators $\mathcal{O}$:

		o^+,	o^-	$\Rightarrow$	o_+,	o_-	
t_i	$\equiv$	$\{T_i\}$,	$\emptyset$	$\Rightarrow$	$\emptyset$,	$\{T_i\}$	
f_i	$\equiv$	$\{F_i\}$,	$\emptyset$	$\Rightarrow$	$\emptyset$,	$\{F_i\}$	
st_i	$\equiv$	$\{T_i, E_0, \ldots, E_m\}$,	$\{S_i\}$	$\Rightarrow$	$\{S_i\}$,	$\emptyset$	
sf_i	$\equiv$	$\{F_i, E_0, \ldots, E_m\}$,	$\{S_i\}$	$\Rightarrow$	$\{S_i\}$,	$\emptyset$	
e_i	$\equiv$	$\emptyset$,	$\{E_i\}$	$\Rightarrow$	$\{E_i\}$,	$\emptyset$	
$pos_{i,j}$	$\equiv$	$\{T_i, E_0, \ldots, E_m\}$,	$\{C_j\}$	$\Rightarrow$	$\{C_j\}$,	$\emptyset$	if $v_i \in c_j$
$neg_{i,j}$	$\equiv$	$\{F_i, E_0, \ldots, E_m\}$,	$\{C_j\}$	$\Rightarrow$	$\{C_j\}$,	$\emptyset$	if $\overline{v_i} \in c_j$.

Assume the following initial and goal state:

$$\begin{aligned} \mathcal{I} &= \{T_1, \ldots, T_m, F_1, \ldots, F_m\} \\ \mathcal{G}_+ &= \{E_0, \ldots, E_m\} \\ \mathcal{G}_- &= \{T_1, \ldots, T_m, F_1, \ldots, F_m\}. \end{aligned}$$

The instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ is, for example, solved by the following plan Δ:

$$\Delta = \langle t_1, \ldots, t_m, f_1, \ldots, f_m, e_0, \ldots, e_m \rangle.$$

Now consider the instance $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ such that

$$\begin{aligned} \mathcal{I}' &= \mathcal{I} \\ \mathcal{G}'_+ &= \{E_0, \ldots, E_m, S_1, \ldots, S_m, C_1, \ldots, C_n\} \\ \mathcal{G}'_- &= \emptyset. \end{aligned}$$

We claim that the SAT formula is satisfiable if, and only if, the plan Δ can be modified by deleting at most m operators and adding some operators before and after the original plan Δ in order to achieve a new plan Δ' that solves Π'.

First, the operators st_i and sf_i can only be added after the original plan because there are $m + 1$ operators e_i at the end of Δ that produce the preconditions for the above operators. Second, in order to achieve the part of the goal specification that requires S_i to hold for each i means that from each pair $\{t_i, f_i\}$ one operator in Δ must be deleted.

Now assume that the SAT formula is satisfiable. In this case, we can delete m of the t_i and f_i operators such that the T_i's and F_i's correspond to a satisfying truth assignment. Then it is trivial to construct a sequence of $pos_{i,j}$'s and $neg_{i,j}$'s that can be added in the end in order to achieve the goal specification requiring C_j, for all $1 \leq j \leq n$, to hold. Conversely, if such a sequence can be found, then the values of T_i and F_i give a satisfying truth assignment for the SAT formula.

Since $st_i, sf_i, pos_{i,j}$, and $neg_{i,j}$ cannot be added before any of the e_i operators, the reduction applies to $\text{MODDELINS}_1^{+,\overline{post}}$, as well.

Membership in NP follows since $\text{PLANSAT}_1^{+,\overline{post}}$ is in NP. Using the same algorithm as described in the proof of Theorem 3 leads to a nondeterministic polynomial-time algorithm for $\text{MODDEL}_1^{+,\overline{post}}$ and $\text{MODDELINS}_1^{+,\overline{post}}$. ∎

We were not able to identify a polynomial planning problem such that the corresponding MODMIX problem becomes NP-complete. The reason for that is that all known polynomial-time planning problems have a particular simple structure. They allow for plans that have the property that if the plan can be extended by adding a set of operators individually, then the plan can be extended by the entire set. Hence, an algorithm for MODMIX would first generate a plan to solve the planning problem instance and then try to extend this plan by as many operators from the old plan as possible.

5 Modifying Plans When the Situations are Similar

The results above could be considered as being not relevant for plan modification in real applications because we made no assumption about the similarity between old and new planning situation. The efficiency gains expected from plan reuse, on the other hand, are based on the assumption that the new situation is *sufficiently close* to the old one—which supposedly permits an easy adaptation of the old plan to the new situation. Beside the fact that this looks like a good heuristic guidance, there is the question whether small differences between the old and the new situation lead to a provable efficiency gain in terms of computational complexity. So it might be perhaps the case that modification is easier than planning if the goal specifications differ only on a constant or logarithmic number of atoms. Although this seems to be possible, there is the conflicting intuition that small changes in the planning situations could lead to drastic (and hard to compute) changes in the plans.

As it turns out, restricting the number of differing atoms does not lead to a different picture than the one presented in the previous section. First of all , Theorem 4 still holds for the restricted versions of the modification problems MODDEL and MODDELINS, where we require the old and new initial states to be identical and the old and new goal specification to differ only on one atom. We call these restricted versions of the modification problem MODDEL1G and MODDELINS1G, respectively.

Theorem 5 *There exists a polynomial-time* PLANSAT$_\rho$ *problem such that the corresponding* MODDEL1G$_\rho$ *and* MODDELINS1G$_\rho$ *problems are* NP*-complete.*

Proof. The transformation used in the proof of Theorem 4 is modified as follows. A new atom B is added, which is assumed to be false in the initial state $\mathcal{I}$ and not mentioned in the old goal specification $\mathcal{G}$. The new goal specification $\mathcal{G}'$ is:

$$\begin{aligned} \mathcal{G}'_+ &= \mathcal{G}_+ \cup \{B\} \\ \mathcal{G}'_- &= \mathcal{G}_-. \end{aligned}$$

Finally, the following operator is added:

$$\{E_0, \ldots, E_m, S_1, \ldots, S_m, C_1, \ldots, C_n\},\ \{B\} \Rightarrow \emptyset,\ \{B\}$$

The MODDEL$_\rho$ and MODDELINS$_\rho$ problems generated by this modified transformation obviously satisfy the constraint that the goal specifications differ only on one atom. Further, the modified transformation has obviously

the same property as the original one, i.e., the generated MODSAT problems can be used to solve the satisfiability problem.

Membership in NP is again obvious. ∎

Although this theorem confirms the intuition that small changes in the goal specification can lead to drastic changes in the plan, it does not rule out the possibility that there are some hard planning problems such that the corresponding modification problems are easy if the goal specification is only changed marginally. In order to rule out this possibility, we would need something similar to Proposition 1. Since there appears to be no general way to reduce PLANSAT$_\rho$ problems to MODSAT1G$_\rho$ problems, we will settle for something slightly less general. We will show that *generating* a plan by modifying a plan for a similar goal specification is at least as hard as the corresponding PLANSAT problem. Hence, instead of the decision problem MODSAT1G, we consider the search problem MODGEN1G. Further, in order to allow for a "fair" comparison between PLANSAT and MODGEN1G, we measure the resource restrictions of MODGEN1G in terms of the size of the planning problem instance—and ignore the size of the old problem.[5] Under these assumptions, the restricted problem MODGEN1G$_\rho$ is always as hard as the corresponding PLANSAT$_\rho$ problem.

Theorem 6 *If* PLANSAT$_\rho$ *is a restricted planning problem that is* PSPACE*-hard or* NP*-hard, then the corresponding* MODGEN1G$_\rho$ *problem is* PSPACE*-hard or* NP*-hard, respectively.*[6]

Proof. Using an algorithm for MODGEN1G$_\rho$, we can *generate a plan* by modifying it iteratively, starting with the empty plan and empty goal specification and continuing by adding step by step one goal atom. Since the size of the goal specification is linearly bounded by the problem instance, we would need only linearly many calls. Supposing that the theorem does not hold would imply that generating a plan under restrictions ρ is easier than PLANSAT$_\rho$, which is impossible by definition. ∎

It should be noted that we did not rely on any particular property of the MODGEN1G$_\rho$ algorithm. In particular, we did not make the assumption that the algorithm has to recycle a maximal reusable plan skeleton. Furthermore, the above theorems apply, of course, also to the modification problems that are restricted to have an one-atom-difference between the initial states.

[5] This is necessary to rule out such pathological situations as the one where modifying an *exponentially* long plan appears to be polynomial while generating it is exponential.

[6] Note that the proof applies to all complexity classes closed under polynomial Turing reductions. Hence, it also applies to the planning problems identified by Erol *et al* [1992]—a fact pointed out to us by Tom Bylander.

6 Discussion

Of course, there arises the question of how the above results relate to practical plan modification systems. Kambhampati and Hendler [1992] investigate plan reuse and modification in the framework of the hierarchical planner and modification system PRIAR, which is based on NONLIN [Tate, 1977]. They use a large number of blocks-world examples in order to evaluate the relative efficiency gains provided by plan modification compared with planning from scratch. In the experiment, the reuse candidate was provided to the planner and thus, no effort for the search in a plan library was spent. The average savings of runtime when plans were reused is given by the authors as 79%.

Hanks and Weld [1992] performed experiments on reusing blocks world plans with their system SPA. This plan generation and modification system is based on a lifted version of McAllester's and Rosenblitt's [1991] systematic nonlinear planning algorithm. In case of the SPA system, the savings turned out to be less drastic than in the PRIAR system. In fact, in the SPA system plan modification can be more expensive than plan generation in terms of runtime if the reuse candidate is not close enough [Hanks and Weld, 1992, p. 103], a situation that did not happen with similar input data in the PRIAR system.

While the relative savings appear to be different for the two approaches, in both cases there is a positive effect which increases when the difference between the new and the old situations decreases. Although this seems to run counter to our complexity results (in particular Theorem 6), these empirical findings do not contradict our results because the experiments were clearly not designed to explore worst-case situations, which complexity analysis is about. An interesting avenue of research would be to characterize the form of planning problems that can exploit plan-reuse techniques to improve the efficiency of the planning process.

What seems to less easily explainable is, however, the discrepancy between the hope that reusing maximal subplans increases the efficiency of plan reuse and our findings. Our results imply that *conservative* plan modification introduces some combinatorics into the planning and reuse process. In particular, as a Corollary of Proposition 2 it follows that is not possible to determine efficiently (i.e., in polynomial time) a maximal reusable plan skeleton *before* plan generation starts to extend the skeleton.

Corollary 7 *It is* PSPACE*-hard to compute a maximal plan skeleton for* MODSAT *instances.*

In other words, plan generation and plan modification cannot be separated. For this reason, the planning process becomes actually more involved when recycling as much of the old plan as possible. Instead of searching for

an arbitrary solution, a plan that contains a maximal subplan of the old plan has to be sought.

Kambhampati and Hendler [1992] mention *conservatism*, i.e., to "salvage as much of the old plan as possible," as an "important desideratum" for a plan modification capability, in order to "ensure efficiency." At a first glance, this seems to be indeed reasonable since it promises to minimize the additional planning effort. As we have seen, however, finding the maximal reusable plan skeleton is already as difficult as planning and is sometimes even more difficult than the corresponding planning problem (Theorem 4). Hence, "conservatism" seems to run counter to increasing planning efficiency.

Having a closer look at the PRIAR framework reveals that plan skeletons are derived in *polynomial time* [Kambhampati and Hendler, 1992, p. 197] by a process called "annotation verification." Hence, by Corollary 7, this process cannot by any means derive maximal applicable plan skeletons. Further, the authors do not give any arguments that they approximate such skeletons. In fact, the skeletons derived by PRIAR are not even guaranteed to be applicable. So, PRIAR does not seem to address the problem of "minimally modifying plans," contrary to what the authors claim.

In fact, *maximal* reuse of an old plan only seems to make sense in a replanning context if costs are charged for *not executing already planned steps*. So, it seems to be the case that the two motivations for plan modification, namely, *replanning* and *reuse* may not be as similar as one might think. While in plan reuse the *efficiency* of the planning process is the most important factor, in *replanning* the minimal disturbance of the old plan may be more important, leading to a more involved planning process.[7]

Plan modification in the PRIAR framework—and in other plan-reuse systems—seems not to be a *computational problem* that has to be addressed, but rather a *solution*, a heuristic technique. The "plan skeleton" that is reused is not the maximal applicable one, but the one that *the particular planning algorithm perhaps can exploit in generating a solution*. In other words, the old plan is used as an "entry point" into the search space of possible plans, as made explicit by Hanks and Weld [1992].

7 Conclusion

Improving the efficiency of planning systems by adding capabilities to modify existing plans has received some research interest recently. In analyzing the computational complexity of this problem, we showed that it is as hard

[7] Kambhampati makes the same distinction in a later paper [Kambhampati, 1992]. Based on arguments concerning the search process of a planner, he also argues that *guaranteeing* that every step that could be reused is reused could be computationally expensive—a conjecture confirmed by Theorem 4.

as planning and *sometimes modification is even harder than planning from scratch.* We showed also that these results hold under the restriction that the modification process has to account for only one changed atom in the goal specification. In particular, we showed that deriving the maximal reusable subplan is not easier than planning. Hence, we cannot hope for minimizing planning effort by first identifying the maximal applicable subplan which is then (minimally) extended by plan generation.

Relating these results to existing plan reuse and modification systems, it turns out that these do not address the modification problem at all, although some authors claim otherwise. In fact, in plan-reuse systems, plan modification is not attacked as a problem but considered as a heuristic technique. This means that instead of "using as much of the old plan as possible" these systems recycle "as much of the old plan as the particular planning algorithm will perhaps be able to use in solving the new problem instance." In fact, adopting the principle of *conservatism* in plan modification only seems to make sense in a replanning context where one wants to minimize the perturbation of the original plan.

Acknowledgements

We would like to thank Christer Bäckström, Tom Bylander, and Subbarao Kambhampati, who provided helpful comments on an earlier version of this paper. In particular, Tom's remarks and questions heavily influenced the paper.

References

[Bäckström and Klein, 1991] Christer Bäckström and Inger Klein. Parallel non-binary planning in polynomial time. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 268–273, Sydney, Australia, 1991.

[Biundo *et al.*, 1992] Susanne Biundo, Dietmar Dengler, and Jana Koehler. Deductive planning and plan reuse in a command language environment. In *Proceedings of the 10th European Conference on Artificial Intelligence, Vienna, Austria*, pages 628–632. John Wiley & Sons, Chichester, UK, 1992.

[Bylander, 1991] Tom Bylander. Complexity results for planning. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 274–279, Sydney, Australia, 1991.

[Bylander, 1992a] Tom Bylander. Complexity results for extended planning. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, Washington, D.C., 1992. Morgan Kaufmann.

[Bylander, 1992b] Tom Bylander. The computational complexity of propositional STRIPS planning. Technical report, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, Ohio State University, Columbus, OH, 1992.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificical Intelligence*, 32(3):333–377, July 1987.

[Chenoweth, 1991] Stephen V. Chenoweth. On the NP-hardness of blocks world. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 623–628, Anaheim, CA, 1991. AAAI Press/The MIT Press.

[Eiter and Gottlob, 1992] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[Erol *et al.*, 1992] Kutluhan Erol, Dana S. Nau, and V. S. Subrahmanian. On the complexity of domain-independent planning. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 381–386, San Jose, CA, 1992. AAAI Press/The MIT Press.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.

[Gupta and Nau, 1991] Naresh Gupta and Dana S. Nau. Complexity results for blocks-world planning. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 629–633, Anaheim, CA, 1991. AAAI Press/The MIT Press.

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

[Hanks and Weld, 1992] Steven Hanks and Daniel 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. Morgan Kaufmann.

[Howe, 1992] Adele E. Howe. Failure recovery analysis as a tool for plan debugging. In *AAAI Spring Symposium on Computational Considerations*

in Supporting Incremental Modification and Reuse, Working Notes, pages 25–30. Stanford University, 1992.

[Johnson, 1990] David S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. A*, pages 67–161. MIT Press, 1990.

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

[Kambhampati, 1992] Subbarao Kambhampati. Utility tradeoffs in incremental plan modification and reuse. In *AAAI Spring Symposium on Computational Considerations in Supporting Incremental Modification and Reuse, Working Notes*, pages 36–41. Stanford University, 1992.

[Koehler, 1992] Jana Koehler. Towards a logical treatment of plan reuse. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, pages 285–286, Washington, D.C., 1992. Morgan Kaufmann.

[McAllester and Rosenblitt, 1991] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 634–639, Anaheim, CA, 1991. AAAI Press/The MIT Press.

[Nebel, 1991] Bernhard Nebel. Belief revision and default reasoning: Syntax-based approaches. In James A. Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 417–428, Cambridge, MA, 1991.

[Selman and Levesque, 1990] Bart Selman and Hector Levesque. Abductive and default reasoning: A computational core. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 343–348, Boston, MA, August 1990.

[Tate, 1977] Austin Tate. Generating project networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 888–893, Cambridge, MA, August 1977.

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

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all publisched papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-91-33
Franz Baader, Klaus Schulz:
Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures
33 pages

RR-91-34
Bernhard Nebel, Christer Bäckström:
On the Computational Complexity of Temporal Projection and some related Problems
35 pages

RR-91-35
Winfried Graf, Wolfgang Maaß: Constraint-basierte Verarbeitung graphischen Wissens
14 Seiten

RR-92-01
Werner Nutt: Unification in Monoidal Theories is Solving Linear Equations over Semirings
57 pages

RR-92-02
Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg:
Π_{ODA}: The Paper Interface to ODA
53 pages

RR-92-03
Harold Boley:
Extended Logic-plus-Functional Programming
28 pages

RR-92-04
John Nerbonne: Feature-Based Lexicons: An Example and a Comparison to DATR
15 pages

RR-92-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Michael Schulte, Rainer Stark:
Feature based Integration of CAD and CAPP
19 pages

RR-92-06
Achim Schupetea: Main Topics of DAI: A Review
38 pages

RR-92-07
Michael Beetz:
Decision-theoretic Transformational Planning
22 pages

RR-92-08
Gabriele Merziger: Approaches to Abductive Reasoning - An Overview -
46 pages

RR-92-09
Winfried Graf, Markus A. Thies:
Perspektiven zur Kombination von automatischem Animationsdesign und planbasierter Hilfe
15 Seiten

RR-92-10
M. Bauer: An Interval-based Temporal Logic in a Multivalued Setting
17 pages

RR-92-11
Susane Biundo, Dietmar Dengler, Jana Koehler:
Deductive Planning and Plan Reuse in a Command Language Environment
13 pages

RR-92-13
Markus A. Thies, Frank Berger:
Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen
13 Seiten

RR-92-14
Intelligent User Support in Graphical User Interfaces:

1. InCome: A System to Navigate through Interactions and Plans
   Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-Oriented User Interfaces
   Markus A. Thies, Frank Berger

22 pages

RR-92-15
Winfried Graf: Constraint-Based Graphical Layout of Multimodal Presentations
23 pages

RR-92-16
Jochen Heinsohn, Daniel Kudenko, Berhard Nebel, Hans-Jürgen Profitlich: An Empirical Analysis of Terminological Representation Systems
38 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka: A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19
Ralf Legleitner, Ansgar Bernardi, Christoph Klauck: PIM: Planning In Manufacturing using Skeletal Plans and Features
17 pages

RR-92-20
John Nerbonne: Representing Grammar, Meaning and Knowledge
18 pages

RR-92-21
Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22
Jörg Würtz: Unifying Cycles
24 pages

RR-92-23
Gert Smolka, Ralf Treinen:
Records for Logic Programming
38 pages

RR-92-24
Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25
Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26
Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitschian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27
Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29
Zhaohur Wu, Ansgar Bernardi, Christoph Klauck: Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

RR-92-30
Rolf Backofen, Gert Smolka
A Complete and Recursive Feature Theory
32 pages

RR-92-31
Wolfgang Wahlster
Automatic Design of Multimodal Presentations
17 pages

RR-92-33
Franz Baader:
Unification Theory
22 pages

RR-92-34
Philipp Hanschke:
Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35
Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36
Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37
Philipp Hanschke: Specifying Role Interaction in Concept Languages
26 pages

RR-92-38
Philipp Hanschke, Manfred Meyer:
An Alternative to H-Subsumption Based on Terminological Reasoning
9 pages

RR-92-41
Andreas Lux: A Multi-Agent Approach towards Group Scheduling
32 pages

RR-92-42
John Nerbonne:
A Feature-Based Syntax/Semantics Interface
19 pages

RR-92-43
Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44
Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45
Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46
Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster: WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

RR-92-48
Bernhard Nebel, Jana Koehler:
Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-51
Hans-Jürgen Bürckert, Werner Nutt:
On Abduction and Answer Generation through Constrained Resolution
20 pages

DFKI Technical Memos

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang:
Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh
A Cognitiv Analysis of Event Structure
21 pages

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
27pages

DFKI Documents

D-92-04
Judith Klein, Ludwig Dickmann: DiTo-Datenbank - Datendokumentation zu Verbrektion und Koordination
55 Seiten

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.): Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.): DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11
Kerstin Becker: Möglichkeiten der Wissensmodel-lierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14
Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-23
Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24
Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25
Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Enno Tolzmann:
Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27
Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages

Plan Modification versus Plan Generation: A Complexity-Theoretic Perspective

Bernhard Nebel, Jana Koehler

RR-92-48
Research Report