Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-52

PHI

A Logic-Based Tool for Intelligent Help Systems

Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul

December 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

PHI - A Logic-Based Tool for Intelligent Help Systems

Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul

DFKI-RR-92-52

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

– PHI –
A Logic-Based Tool for Intelligent Help Systems

M. Bauer S. Biundo D. Dengler J. Koehler G. Paul

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
e-mail: {last-name}@dfki.uni-sb.de

Abstract

We introduce a system which improves the performance of intelligent help systems by supplying them with plan generation and plan recognition components. Both components work in close mutual cooperation. We demonstrate two modes of cross-talk between them, one where plan recognition is done on the basis of abstract plans provided by the planner and the other where optimal plans are generated based on recognition results. The examples which are presented are taken from an operating system domain, namely from the UNIX mail domain.

Our system is completely logic-based. Relying on a common logical framework—the interval-based modal temporal logic LLP which we have developed—both components are implemented as special purpose inference procedures. Plan generation from first and second principles is provided and carried out deductively, whereas plan recognition follows a new abductive approach for modal logics. The plan recognizer is additionally supplied with a probabilistic reasoner as a means to adjust the help provided for user-specific characteristics.

Contents

1 Introduction

Intelligent help systems aim at providing advanced active help to the users of complex software systems (cf. [Bre90, TB92, NWW93]). The performance of these help systems can be considerably improved if they are supplied with *plan recognition* and *plan generation* capabilities. Observing a user and recognizing his goals enables the system to help by taking into account the current state of the system as well as the user's level of education and current behavior. Moreover, if a planning capability is available support can be given by proposing appropriate (sub-)plans or even by executing them automatically. In particular, this improves the assistance provided if the planning component can rely upon any observations and plan recognition results; that is exactly what PHI aims to achieve.

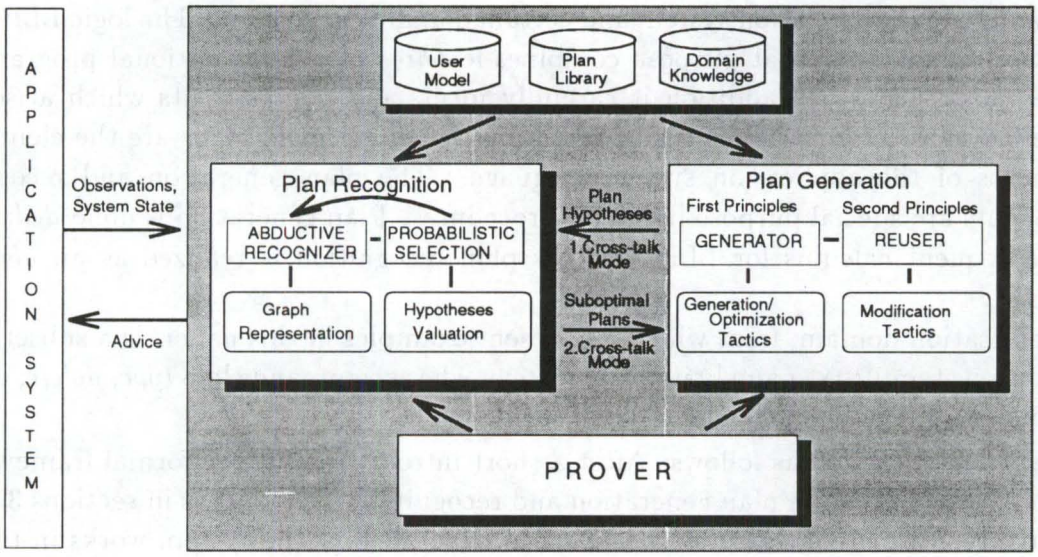

Figure 1: PHI: System Architecture

PHI (cf. figure 1), the system being presented in this paper, is a tool for intelligent help systems. It provides both a plan recognizer and a planning component and one of its main characteristics consists in the close mutual cooperation between the two components. There are several *cross-talk modes*. The first one is devoted to realizing plan recognition on the basis of *abstract* plans produced by the planner. Abstract plans are those which represent a variety of "concrete" observable action sequences by admitting several degrees of freedom like *variables* (abstracting from the objects involved), *abstract commands* (abstracting from the names of actions which have the same effects), or *temporal abstraction* (abstracting from the point in time at which an action occurs). The generation of plans is based on standard assumptions concerning goals that typically occur or are specific to a certain user. Abstract plans are generated from these formal plan specifications. In doing so, the planner not only performs planning from first principles but is able to *reuse* already existing plans which are stored in a library (planning from second principles). The plans provided serve as plan hypotheses in the recognition process. Taking abstract plans instead of concrete ones keeps the hypothesis space of manageable size. The plan hypotheses are passed to the recognition component where they are provided with numerical values which reflect the probabilities of their being confirmed by the subsequent observations. These a priori probabilities mirror a specific user's behavior, and are taken

from the user model. Having observed the user's actions step by step the plan recognizer consequently tries to confirm the plan hypotheses by proving that the action sequence observed up to now is an admissible "instance". Hypotheses which are not confirmed are rejected and with that the probability distribution of the hypothesis space changes dynamically.
In the first cross-talk mode the plan recognizer is able to determine the most likely plan a user follows by carrying out appropriate "instantiations" on valid plan hypotheses. In addition, services like *semantic plan completion* can be offered at any time during the observation process.
The second cross-talk mode is devoted to providing the user with *optimal* plans whenever suboptimal behavior has been recognized or aid has explicitly been sought.

The system is completely *logic-based.* It requires a proper axiomatization of the basic commands of the application system and certain domain constraints. The logic LLP which we have developed for that purpose combines features of both traditional programming and temporal logics. In addition it carefully meets any requirements which arise from this help system context, since the basic actions which occur in plans are the elementary statements of the application system language. The plan generation and recognition components are special purpose inference procedures. Plan generation is done *deductively* using a sequent calculus for LLP, whereas plan recognition is realized as an *abductive* process.
The application domain, from which we present examples in this paper, is a subset of the operating system UNIX, namely its *mail system*, where commands like *type*, *delete*, or *save* manipulate objects like *messages* or *mailboxes.*

The paper is organized as follows: After a short introduction to the formal framework in section 2 we describe the plan generation and recognition components in sections 3 and 4, respectively. We demonstrate by means of an example how the system works in the first cross-talk mode. In section 5 we briefly describe the implementation of our system and finally we conclude with some remarks in section 6.

2 The Formal Framework

Plan generation and plan recognition are carried out on a common logical basis. The *logical language for planning* (LLP) [BD93], which we have developed for this purpose, combines features of *Choppy Logic* [RP86] with the *Temporal Logic for Programs* [Krö87]. This entails the consideration of plans as programs as has also been proposed by other authors (cf. [Gre69], [Bib86], [MW87], [FK91]). This view fits well into our help system context as the plans we have to generate and recognize consist of the application system's elementary statements and may even contain *control structures* like conditionals or loops. LLP is an interval-based modal temporal logic. It provides the modal operators $\bigcirc$(next), $\Diamond$ (sometimes), $\Box$ (always), and the binary modal operator ; (chop), which expresses the sequential composition of formulas. Besides these operators control structures are also available, as in programming logics. Basic actions, which in our example domain are the elementary mail commands, are axiomatized like assignment statements in programming logics. The state changes which they perform are reflected in changing the values of

certain variables. The *type* command, for example, reads:

$$\forall x\,[\,[open\_flag(mb) = T \wedge d\_flag(x, mb) = F \wedge EX(type(x, mb))\,] \rightarrow \\ \bigcirc r\_flag(x, mb) = T\,]$$

It states that if a certain message x in a current mailbox mb has not yet been deleted and we execute the *type* command then the message is read in the next state.
Plans are represented by a certain class of LLP formulas. Besides basic actions (expressed by the *execute* predicate EX) they contain the *chop* operator, control structures, and also temporal abstractions.
Plan specifications are LLP formulas of the form as follows:

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

i.e., if the *preconditions* hold in a situation where we carry out Plan, then the *goals* will be reached. For example the following formula

$$open\_flag(Mbox) = T \wedge \mathsf{Plan} \rightarrow \\ \Diamond[display = allheaders(Mbox) \wedge \Diamond\,[r\_flag(x, Mbox) = T\,]\,] \tag{1}$$

specifies the plan "Display the content of mailbox $Mbox$ on the screen and then read message x". Plan is a metavariable for a plan formula.
Plan generation (cf. section 3) is carried out by *constructively* proving the specification formula. While proving the specification formula the plan metavariable Plan is replaced by a plan (formula) which satisfies the specification. Proving (1), for example, results in the following plan:

$$EX(header(Mbox))\ ; \\ \textbf{if}\ d\_flag(x, Mbox) = T\ \textbf{then}\ EX(undelete(x, Mbox)); \\ EX(type(x, Mbox)) \tag{2}$$

As the UNIX mail language does not of course provide any control structures, this plan has to be considered as abstract, apart from the fact that the variable x occurs in it. In section 4 we describe the methods which, for example, in a certain situation serve to recognize the observed action sequence

$$EX(header(Mbox)); EX(type(2, Mbox)) \tag{3}$$

as an admissible "instance" of (2).
In section 3 we will see how the conditional plan above is generated from first principles. We can then demonstrate planning from second principles by reusing and modifying this plan so that it meets a new specification:

$$open\_flag(Mbox) = T \wedge d\_flag(x, Mbox) = F \wedge \mathsf{Plan} \rightarrow \\ \Diamond[display = allheaders(Mbox) \wedge \\ \Diamond\,[r\_flag(x, Mbox) = T \wedge \Diamond[d\_flag(x, Mbox) = T]\,]\,] \tag{4}$$

We obtain

$$EX(header(Mbox)); EX(type(x, Mbox))\,; EX(delete(x, Mbox)) \tag{5}$$

as a result. In a plan recognition example (cf. section 4) this plan can then serve as one of the plan hypotheses.

3 Plan Generation

The planning system (cf. [BDK92]) works by using techniques of planning from both first and second principles. Planning from first principles begins with a plan specification. The plan is generated on the basis of the *domain knowledge* provided. Planning from second principles adds the ability to incorporate previously generated plans and the problem solving knowledge obtained thereby.
In the first cross-talk mode, abstract plans are generated in order to provide the plan recognizer with plan hypotheses. To generate these hypotheses, the planner works from second principles by reusing formerly generated plans. In the second cross-talk mode, optimal or user-satisfactory plans are generated from first principles.

3.1 Planning from First Principles

By using a sequent calculus for LLP (cf. [BD93]) the plan generator tries to find a constructive proof for the plan specification formula so that an instantiation for the plan metavariable can be obtained. We thus have a plan the execution of which is sufficient to reach the goals specified, i.e., a plan which meets the specification. Following the paradigm of *tactical theorem proving* (cf. [Con86], [HRS90], [Pau90]) the proof is guided by special *planning tactics* written in a metalogical tactic language.
As for plan specification (1), the proof is carried out by dividing the specification formula into subformulas, i.e., those representing single subgoals which the plan has to reach. We can simultaneously introduce a structure into the plan metavariable Plan, which states that Plan should consist of at least two subplans: $\mathsf{Plan} \equiv \mathsf{P}_1 \,;\mathsf{P}_2$.
Let us now consider the generation of a plan for P_2. The corresponding subgoal reads:

$$open\_flag(Mbox) = T \wedge \mathsf{P}_2 \rightarrow \Diamond r\_flag(x, Mbox) = T$$

Usually subgoals of this type are proven by using nonlogical axioms which describe basic actions. Thus, the plan metavariable is instantiated by a basic plan formula. The instance of the *type* axiom below is selected because it can reach the desired goal of setting the r_flag to T.

$$open\_flag(Mbox) = T \wedge d\_flag(x, Mbox) = F \wedge EX(type(x, Mbox))$$
$$\rightarrow \circ r\_flag(x, Mbox) = T$$

The preconditions of this action however must hold in order to make the axiom applicable. One of these preconditions is missing from the subgoal above. One strategy used in order to establish such a precondition is to derive it from the facts which hold after P_1 has been executed. In our case this strategy fails. Following a deductive version of the *means-ends analysis* (cf. [FN71], [Nil80]) we therefore introduce an additional subplan which produces the missing precondition. Thus, P_2 becomes the composition of a one-armed conditional and a subplan P_4, respectively:

$$\mathsf{P}_2 \equiv [\textbf{if } d\_flag(x, Mbox) = T \textbf{ then } \mathsf{P}_3];\mathsf{P}_4$$

The new subgoal obtained is:

$$open\_flag(Mbox) = T \wedge \textbf{if } d\_flag(x, Mbox) = T \textbf{ then } \mathsf{P}_3$$
$$\rightarrow \circ[open\_flag(Mbox) = T \wedge d\_flag(x, Mbox) = F]$$

To properly instantiate P_3 an instance of the *undelete* action axiom can be used; this tells us that the execution of $undelete(x, Mbox)$ makes $d\_flag(x, Mbox) = F$ true in the next state, should it not have held before. In a similar way P_4 can now be instantiated by using the *type* action axiom.
The overall plan which results after the proof tree has been completed and all plan metavariables have been instantiated, is the plan given by formula (2) above. It clearly meets the specification in (1).
In addition to subgoals whose proof leads to instantiations of the plan metavariables, as in the above examples, so-called *plan assertions* must also be proven. These represent certain properties which are required by the plan to be generated. A typical example in our case is the fact that the formula $open\_flag(Mbox) = T$—which acts as a precondition to the whole plan—does survive the execution of subplan P_1.
In our system, planning from first principles is, like several other approaches to deductive planning (cf. [Gre69], [Bib86], [MW87], [FK91]) closely related to work done on *deductive program synthesis* where *programs* are generated by proofs (cf. [MW80], [Fra88], [HRS91], [Biu92]).

3.2 Planning from Second Principles

By supplying a reuse component to the deductive plan generator we pursue two main goals: Firstly, the *efficiency* of planning is to be improved by avoiding the repetition of the same planning effort. Secondly, *flexibility* is added to planning by incorporating knowledge concerning previously used problem solving strategies.
The reuser first takes the current specification and searches in a *plan library* for a plan which can be reused as its solution. The plan library is organized in the form of a hierarchical network of *plan entries*. Each plan entry contains (a) the specification of the planning problem, (b) the plan which was generated as a solution, and (c) an annotation which stores information which was extracted from the generation process, e.g., how the actions occurring in the plan are related to certain specification subgoals etc. This plan library is created and updated dynamically when plans are generated.
The current specification guides the search process in the plan library as it is the only source of information available when the reuse process must begin. Current preconditions and goals are identified and a plan entry is searched for in which similar preconditions and goals occur. If a plan entry does not meet the search criteria, planning from first principles must begin.
If the search in the plan library terminates successfully with a plan entry, the reuser must verify if the plan stored in this entry can provide a solution to the current planning problem. The verification is carried out by a formal proof in which the *prover* verifies that at least the preconditions the plan requires hold in the current situation and that at most the goals achieved by the plan are required as current goals.
If the proof succeeds, the plan provides a provably sound solution to the current planning problem; if it fails, the plan has to be modified.
The *modification tactics* analyze the failed proof and modify the plan using information stored in the annotation.
Let us assume, for example, that specification 4

$$open\_flag(Mbox) = T \wedge d\_flag(x, Mbox) = F \wedge \mathsf{Plan} \rightarrow$$
$$\Diamond[display = allheaders(Mbox) \wedge$$
$$\Diamond\,[r\_flag(x, Mbox) = T \wedge \Diamond[d\_flag(x, Mbox) = T]]]$$

is given to the planner in the first cross-talk mode. Planning from second principle starts and tries to reuse plan (2).

Comparing it with specification (1) it is obvious that more preconditions are given, but even more goals are required in (4). In this case, the prover reports a failure because more goals are required in specification (4) than are achieved by the plan. The modification tactic identifies the missing subgoal $d\_flag(x, Mbox) = T$ for which a subplan has to be generated from first principles. Furthermore, it has to inspect the temporal structure of the plan to be reused in order to determine the point in time at which this subplan has to be inserted. For this purpose, explicit representations of the temporal models of both specifications are constructed and compared during the proof.

Besides the modifications that are necessary in obtaining a solution to the current specification, the tactic extracts information from the proof for a subsequent optimization of the plan. An optimization in the context of plan reuse means, e.g., to remove superfluous actions from the reused plan. A plan is a solution if it achieves at least all the goals that are required in the current specification, i.e., if the plan achieves some additional subgoals it is still considered to be a solution. In some applications however, plans have to be minimal in the sense of achieving exactly the goals required. The plan reuse component is able to perform the necessary optimizations in these cases.

In the example, the reuser detects that the case analysis in the reused plan is superfluous because the condition on which it depends is explicitly given in the specification. Therefore, the conditional can be deleted from the plan. The result of the modification process is a *plan skeleton* for a sequential plan

$$EX(header(Mbox)); EX(type(x, Mbox)); \mathsf{Plan}_1 \qquad (6)$$

containing the reusable subplan identified during the proof and a meta variable Plan_1 as a "placeholder" for the completing subplan which has to be generated in order to reach the additional goal.

The generator uses the plan skeleton as a partial instantiation of the plan metavariable Plan in specification (4). This simplifies the constructive proof of the specification: The partial proof tree for which an instantiation of the metavariable is already known can be easily expanded without further search effort. To replace the metavariable Plan_1 occurring in the skeleton, the generator has to plan from first principles leading to the instantiation $EX(delete(x, Mbox))$. The interleaving of proof tree reconstruction and generation ensures that the modified plan provides a provably sound solution to the current plan specification that can be sent to the plan recognizer as a plan hypothesis.

The approach we follow investigates plan reuse in the general context of deductive planning and has been described in more detail in [BDK92, Koe92]. Other current approaches investigate plan reuse and modification in the framework of classical STRIPS-like planners, e.g., the hierarchical planner and modification system PRIAR [KH92], wich is based on NONLIN [Tat77], or in the framework of case-based reasoning, e.g., the systems SPA [HW92] or CHEF [Ham90].

A complexity-theoretic analysis studying plan modification vs. plan generation can be found in [NK92].

3.3 Generating Optimal Plans

The second cross-talk mode is concerned with the generation of optimal and user-satisfactory plans. The generator receives a plan specification which either belongs to a plan recognized as suboptimal by the plan recognition component or is derived from a request for passive help.
Planning in this mode is based on a dynamically changing adjustment of the generation process triggered by *plan quality criteria* derived from the user model. The generator considers, e.g., the user's preferences, his knowledge about the domain, and his typical behavior in order to generate satisfactory plans for him. It produces a user-adapted concrete plan that meets the specification and is as short as possible according to the number of basic actions used. Since planning is done deductively the adjustment essentially places a restriction on the sets of nonlogical axioms and rules.
If, on the basis of the current plan quality criteria, no plan can be found, then the criteria must be minimally changed in order to generate a plan. The necessary deviations are recorded and can be used by a tutorial system to teach the user accordingly. In the case of a recognized suboptimal plan, the generated optimal plan is, e.g., the basis for an active user support of the help system. Generation of optimal plans is only carried out from first principles because the reuse of concrete plans requires consideration of dynamically changing plan quality criteria which can contradict the aim of making planning more efficient.

4 Plan Recognition

The recognition of plans in this logic-based context is realized by a *generalized abductive process* with a *probabilistic valuation* of hypotheses (cf. figure 1). Starting from plan hypotheses synthesized by the plan generation component and observations of user actions, an attempt is made to identify an hypothesis describing the user's pursued plan. The use of probabilistic reasoning allows us to determine *one* "best" hypothesis if, e.g., semantic plan completion is to be offered.

4.1 The Abductive Recognizer

Plan recognition, which is the identification of a user's behavior given an observed goal or action, can be viewed as an inherently abductive problem, if a plan hypothesis P is interpreted as an assumption explaining the observed action a, i.e., $\mathcal{T} \cup \{P\} \models a$, where $\mathcal{T}$ describes the domain knowledge (e.g., [AP90], [Sha89], [HK90], [Wær92]). P is required to be a *ground instance* of an element of a set of predefined candidate explanations called *abducibles.*[1]
However, this "classical" abduction principle is insufficient for temporal hypotheses incorporating an implicit representation of time, because the correctness criterion is no longer satisfied: For example, we cannot deduce from the hypothesis $\Diamond EX(a)$—a is expected to be executed at some time—that a is executed now, i.e., the fact $EX(a)$. Nevertheless this is an intuitively valid hypothesis since anticipating an action at some time might

[1] For an introduction to abduction see, for example, [Pei58] or [Fan70]. An overview can be found in [KKT92] and [Mer92].

indeed be an explanation for its present occurrence. To overcome this deficiency the abductive process is divided into two phases: 1. A *guessing phase* where modal hypotheses are adopted—these abducibles are the abstract plans provided by the plan generation component—and 2. a *validation phase* where the hypotheses are verified with respect to the sequence of observed actions. This is the task of plan recognition.
In addition, the ground instance requirement has to be adapted to our modal logic context. It is generalized in such a way that hypotheses are made more precise not only by instantiating variables but in a temporal sense as well. We *concretize* the hypotheses according to the observations made in each recognition step.
This process of concretizing will be explained using a short example with two observation steps. It is assumed we are given the following plan hypothesis synthesized by the generation component as described in section 3.

$$P1 = EX(header(Mbox)); EX(type(x, Mbox)); EX(delete(x, Mbox))$$

Together with this plan formula the plan recognizer is given a set of preconditions which stem from plan specification (4): $\{open\_flag(Mbox) = T \wedge d\_flag(x, Mbox) = F\}$. These preconditions ensure the applicability of the generated plan and thus have to hold for a valid concrete plan instance. If they cannot be proven in the given recognition scenario the corresponding plan has to be rejected. For the sake of simplicity we will omit them here.
Suppose we have the additional hypotheses

$$P2 = \Diamond EX(header(Priv)); EX(type(x, Priv)); EX(delete(x, Priv))$$
$$P3 = \Diamond EX(header(Priv)); EX(next(x, Priv)); EX(delete(x, Priv))$$

i.e., contrary to the first hypothesis the *header* command is not expected to be the first action but may also occur at some later stage. In addition, a mailbox named *Priv* is supposed to be the current one here. For both hypotheses we also ignore the preconditions. Observing the user execute the action $EX(folder(Priv, Mbox))$, i.e., he moves from the mailbox *Mbox* to the mailbox *Priv*, leads to the following concretized hypotheses:

$$P1^1 = false$$
$$P2^1 = EX(folder(Priv, Mbox)) \wedge \circ\Diamond EX(header(Priv)); EX(type(x, Priv)); EX(delete(x, Priv))$$
$$P3^1 = EX(folder(Priv, Mbox)) \wedge \circ\Diamond (EX(header(Priv)); EX(next(x, Priv)); EX(delete(x, Priv))$$

Concerning $P1$ we definitely anticipated $EX(header(Mbox))$. This cannot be concretized with the first observation and thus the hypothesis has to be rejected. In $P2$ and $P3$ the initially expected user action is not specified, we only know that the given action sequence is executed at some time. So, after observing an action not matching the first expectation, we know that this sequence cannot start before the next observation, i.e., we get as parts of the new hypotheses $P2^1$ and $P3^1$

$$\circ\Diamond EX(header(Priv)),$$

the action *header* is anticipated at some future time and not before the next state.
If the second observation is $EX(header(Priv))$, the concretization of both $P2$ and $P3$ is a disjunction of two hypotheses. Either we recognize the *header* command in the "sometimes" sequence, or we make a decision to expect it later. Thus, we have

$$P2_1^2 = EX(folder(Priv, Mbox)) \wedge \circ EX(header(Priv)); EX(type(x, Priv)); EX(delete(x, Priv))$$

$$P2_2^2 = EX(folder(Priv, Mbox)) \wedge \circ EX(header(Priv)) \wedge \circ\circ\Diamond EX(header(Priv)); EX(type(x, Priv)); EX(delete(x, Priv))$$

As the hypotheses for $P3$ are similar we will omit them here.
A hypothesis is said to be *recognized* if the sequence of observations implies its concretization. Considering the example, this is the case for $P2$ if, e.g., the next observations are $EX(type(2, Priv))$ and $EX(delete(2, Priv))$.
Besides the theoretical foundations, a method was developed for computing concretized explanations. An algorithm solving this problem must be able to identify at each state of time the part of the considered hypotheses being affected at that moment. To do this in an efficient way and without using explicit reasoning over modal operators in each iteration step, we use a *transformation* of LLP formulas into graphs that contain solely first-order formulas.[2] This transformation is carried out once before starting recognition. The plan recognition process is realized by moving through the graph according to the observations made. A theorem prover is used to deduce relations between abstract commands in the hypothesis and observed actions. Preconditions are tested by inquiring the application about the current state of the system.
Several properties of this generalized abduction principle, e.g., the connection to "classical" abduction and the relationship between the original and concretized hypotheses have been proven. In addition, a proper semantics has been given. For details see [Pau92].
The method described above allows us to recognize temporal abstractions as well as abstract plans containing LLP control structures such as conditionals and loops. We are able to retransform the graph in each iteration step, thus obtaining a history of the actions observed and a description of the expected continuation of the plan. By that we are able to offer semantic plan completion at any time as long as valid hypotheses are available. Probabilistic selection (cf. section 4.2) is the method used in determining the "best" hypothesis for that purpose.

4.2 Probabilistic Selection

The plan recognizer described so far manipulates *sets* of plan hypotheses each of which is considered equally plausible. If, however, a decision for *one* alternative is required (e.g., to offer semantic plan completion to the user at an early stage), we must be able to determine the "best" choice among them. To do so, we can exploit knowledge about user preferences stored in a kind of user model and encoded in a probabilistic mechanism.
Probabilistic reasoners (e.g., [KSH91]) use a knowledge base as a back-up, which contains the set of all possible hypotheses together with numerical values assigned to them somehow representing their specific probability. These numbers usually stem from long-term observations of the domain and from statistics. The situation in the first cross-talk mode, however, is somewhat different because the search space of the plan recognizer—the set of all plan hypotheses—is generated dynamically. So, we cannot expect to have any statistical information at hand that directly applies to it. Yet, we can evaluate the user behavior observed according to criteria like typical action sequences, frequently pursued goals, etc.

[2] Graph- or so-called tableau-based methods are used for a variety of temporal logics. See, for example, [Pra79], [BAHP82], and [Wol85].

If we know, for example, that the user tends to delete a message immediately after reading it, and that he usually prefers the *type* command to *next*, we might come up with a set C of LLP formulas similar to abstract plans and formulas encoding desired system states (cf. section 2) describing the user's default behavior. Each formula is assigned a numerical value representing the statistical information about it. In the example above, C might include the following entries

$$\begin{aligned} C_1: &\quad \langle\ \exists x, mb.\Diamond(EX(type(x,mb)); EX(delete(x,mb))),\ 0.4\ \rangle \\ C_2: &\quad \langle\ \exists x, mb.\Diamond(EX(next(x,mb)); EX(delete(x,mb))),\ 0.2\ \rangle \\ C_3: &\quad \langle\ \exists x, mb.\Diamond(EX(read\_mail(x,mb)); EX(delete(x,mb))),\ 0.1\ \rangle \\ &\quad \vdots \end{aligned}$$

The entry C_3 here means that for a small part of all observed cases, we know only that the user executed a *type or* a *next* command to read his messages, expressed by using the abstract command *read_mail*. The formula in C_3 is more general than those of C_1 and C_2. Such relations induce a hierarchical structure on C that is exploited during the numerical computations.

The numerical values distributed among the members of C sum up to 1 and form a *mass distribution* from Dempster-Shafer theory (DST) (cf. [Sha76], [SP90]). While interpreting the value 1 as perfect certainty, smaller numbers represent the degree of partial confidence we might have in the validity of the various propositions.

From a mass distribution m, we can derive the so-called *belief* and *plausibility* functions Bel_m and Pl_m, respectively. These two values make up a probability interval stating that the "true" probability of some proposition A lies somewhere between $Bel_m(A)$ and $Pl_m(A)$, but cannot be determined on the basis of the knowledge at hand. Thus, we are able to express *partial ignorance*.[3]

Classifying the plan hypotheses obtained from the plan generator according to the formulas C_i in C, a set of valuated hypotheses sets PH_{C_i} is obtained each of which inherits the numerical value originally attributed to its classification criterion in C. In our example, P_1 and P_2 become members of the class PH_{C_1}, P_3 becomes an element of PH_{C_2}. In addition, they are all placed in class PH_{C_3}. Thus the original hierarchy of classification criteria according to generality mentioned above mirrors itself in the subset/superset relation of the associated hypotheses sets that make up the *plan hierarchy PH*.

After this preprocessing, the probabilistic selection module in every recognition step obtains the most recent observation together with information about those hypotheses which are no longer valid. On the basis of this knowledge, we can compute an updated mass distribution on PH reflecting the impact of the new observation on the a-priori valuation of the hypotheses. Dempster's rule is the basis for this computation which is explained in more detail in [Bau92].

Let us assume we are given the information that P_1 is no longer valid. Then, the observation $EX(header(Priv))$, for example, may lead to a new mass distribution where $PH_{C_1} = \{P_2\}$ and $PH_{C_2} = \{P_3\}$ are attributed the values 0.5 and 0.35, respectively, and $PH_{C_3} = \{P_2, P_3\}$ is valuated with 0.15. If we recall from DST that the *belief* in a set A of hypotheses is computed by summing up the respective mass values attributed to all of its subsets and the *plausibility* is the result of adding the mass values of all sets having a non-empty intersection with A, we can interpret these numbers as follows: Currently,

[3] For an examination of the relations between mass distributions, probabilities, and DST see [KSH91].

hypothesis P_2 appears to be the most likely because its probability lies somewhere in the interval $[0.5, 1]$. If semantic plan completion is required, P_2 is the current offer of the system to the user. If we are forced to additionally provide a hypothesis *definitely* containing the plan pursued by the user, we can choose the disjunction of P_2 and P_3 because the set PH_{C_3} which contains these only is the smallest set with attributed belief 1, i.e., it represents the most specific hypothesis we *certainly* believe in. This property uniquely determines the disjunction of P_2 and P_3 because all other plan hypotheses that might be contained in PH are attributed mass 0 and thus are considered impossible according to the evidences obtained so far.

5 Implementation

A prototype of the PHI system has been implemented in SICSTUS PROLOG on a SUN SPARC computer. It emerged that the deductive planning methods in their running time behavior correspond to conventional knowledge based planning. This is reached by using proof strategies that are especially tailored to the planning tasks to be performed. These strategies skilfully restrict the search space and keep the backtracking rate low.
The performance of the plan recognizer is improved by using an equivalence-preserving graph-based representation of plan formulas and efficient algorithms working on them.
Therefore, our logic-based approach seems to be well suited even for real help systems.

6 Conclusion

A new approach in implementing intelligent help systems has been introduced. It is based on a logic developed especially for the command language environments in which help systems are embedded. Plan recognition and plan generation components are special purpose inference processes. They work in close mutual cooperation (cross-talks). One cross-talk mode is devoted to plan recognition on the basis of abstract plans provided by the planner; another mode works on generating optimal plans on the basis of recognition results.
Planning from first as well as from second principles is done deductively combining ideas borrowed from the logic-based treatment of programs with those of tactical theorem proving. The resulting plans are provably sound w.r.t. their specifications. Plan recognition is based on a new abductive principle for modal logics. The recognizer is additionally supplied with a probabilistic reasoner, a means to improve the help provided by taking into account user-specific characteristics as well as general heuristics.
Realizing plan generation and recognition in such a strictly logic-based way nevertheless does not cause any considerable inefficiencies, even for "real" *mail* plans that are much more complex than the examples shown in this paper. This suggests evaluating the PHI approach even for richer application domains.

References

[AP90] D.E. Appelt and M. Pollack. Weighted abduction for plan ascription. Technical report, Artificial Intelligence Center and Center for the Study of Language and Information, SRI International, Menlo Park, California, 1990.

[BAHP82] M. Ben-Ari, J.Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity, and completeness. *Comput. System Sci.*, 25:402–417, 1982.

[Bau92] M. Bauer. Plan recognition under uncertainty. DFKI Research Report, German Research Center for Artificial Intelligence, 1992. to appear.

[BD93] S. Biundo and D. Dengler. The logical language for planning LLP. DFKI Research Report, German Research Center for Artificial Intelligence, 1993. to appear.

[BDK92] 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*, pages 628–632, 1992.

[Bib86] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[Biu92] S. Biundo. *Automatische Synthese rekursiver Programme als Beweisverfahren.* Springer IFB 302, Berlin, Heidelberg, New York, 1992.

[Bre90] J. Breuker. *EUROHELP Developing Intelligent Help Systems.* EC, Kopenhagen, 1990.

[Con86] R.L. Constable. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, 1986.

[Fan70] K.T. Fann. *Peirce's Theory of Abduction.* Martinus Nijhoff, The Hague, 1970.

[FK91] M. Franova and Y. Kodratoff. Solving "How to Clear a Block" with constructive matching methodology. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 232–237, 1991.

[FN71] R.E. Fikes and N.J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Fra88] M. Franová. *Fundamentals of a New Methodology for Program Synthesis from Formal Specifications: CM-Construction of Atomic Formulae.* Ph.D. Thesis, Université de Paris-Sud, Orsay, 1988.

[Gre69] C. Green. Application of theorem proving to problem solving. In *Proceedings of the 2th International Joint Conference on Artificial Intelligence*, pages 219–239, 1969.

[Ham90] K. J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45:173 – 228, 1990.

[HK90] N. Helft and K. Konolige. Plan recognition as abduction and relevance. Draft version, Artificial Intelligence Center, SRI International, Menlo Park, California, 1990.

[HRS90] M. Heisel, W. Reif, and W. Stephan. Tactical theorem proving in program verification. In *Proceedings of the 10th Conference on Automated Deduction*, pages 117–131. Springer LNCS 449, 1990.

[HRS91] M. Heisel, W. Reif, and W. Stephan. Formal software development in the kiv system. In *Automating Software Design, R. McCarney and M.R. Lowry (eds.)*. AAAI Press, 1991.

[HW92] 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. Morgan Kaufmann, Menlo Park.

[KH92] S. Kambhampati and J. A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55:193 – 258, 1992.

[KKT92] A.C. Kakas, R.A. Kowalski, and F. Toni. Abductive logic programming. Draft version, Department of Computer Science, University of Cyprus, Nicosia, and Imperial College of Science, Technology and Medicine, London, 1992.

[Koe92] J. 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, Menlo Park.

[Krö87] F. Kröger. *Temporal Logic of Programs.* Springer, Heidelberg, 1987.

[KSH91] R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge Based Systems.* Springer, 1991.

[Mer92] G. Merziger. Approaches to abductive reasoning – an overview. *Artificial Intelligence Review*, 1992. to appear.

[MW80] Z. Manna and R. Waldinger. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, 2:90–121, 1980.

[MW87] Z. Manna and R. Waldinger. How to clear a block: Plan formation in situational logic. *Journal of Automated Reasoning*, 3:343–377, 1987.

[Nil80] N. J. Nilsson. *Principles of Artificial Intelligence.* Springer, New York, 1980.

[NK92] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. DFKI Research Report RR-92-48, German Research Center for Artificial Intelligence, 1992.

[NWW93] P. Norwig, W. Wahlster, and R. Wilensky. *Intelligent Help Systems for UNIX - Case Studies in Artificial Intelligence.* Springer, Heidelberg, 1993. to appear.

[Pau90] L. Paulson. Isabelle: The next 700 theorem provers. In P. Odifredi, editor, *Logic and Computer Science.* Academic Press, 1990.

[Pau92] G. Paul. A generalized abductive principle for a modal temporal logic. DFKI Research Report, German Research Center for Artificial Intelligence, 1992. to appear.

[Pei58] C.S. Peirce. *Collected Papers of Charles Sanders Peirce (eds. C. Hartshorne et al.)*. Harvard University Press, 1931-1958.

[Pra79] V. Pratt. Models of program logics. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, pages 115–122, 1979.

[RP86] R. Rosner and A. Pnueli. A choppy logic. In *Symposium on Logic in Computer Science*, Cambridge, Massachusetts, 1986.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, 1976.

[Sha89] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1055–1060, 1989.

[SP90] G. Shafer and J. Pearl, editors. *Readings in Uncertain Reasoning.* Morgan Kaufmann Publishers, Los Altos, California, 1990.

[Tat77] A. Tate. Generating project networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 888–893, Cambridge, MA, 1977. Morgan Kaufmann, Menlo Park.

[TB92] M.A. Thies and F. Berger. Plan-based graphical help in object-oriented user interfaces. In *Proceedings of the International Workshop on 'Advanced Visual Interfaces'*, Rome, Italy, May 1992.

[Wær92] A. Wærn. Reactive abduction. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 159–163, 1992.

[Wol85] P. Wolper. The tableau method for temporal logic: an overview. *Logique at Anal.*, 28:119–136, 1985.

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-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 Θ-Subsumption Based on Terminological Reasoning
9 pages

RR-92-40
Philipp Hanschke, Knut Hinkelmann: Combining Terminological and Rule-based Reasoning for Abstraction Processes
17 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-92-47
Frank Bomarius: A Multi-Agent Approach towards Modeling Urban Traffic Scenarios
24 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

RR-92-52
Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul:
PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-54
Harold Boley: A Direkt Semantic Characterization of RELFUN
30 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-92-06
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

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

PHI - A Logic-Based Tool for Intelligent Help Systems

Mathias Bauer, Susanne Biundo, Dietmar Denger, Jana Koehler, Gabriele Paul

RR-92-52
Research Report