PHI—A Logic-Based Tool for Intelligent Help Systems *

M. Bauer and S. Biundo and D. Dengler and J. Koehler and G. Paul
German Research ('enter for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbriicken
e-mail: {lastname}@dfki.uni-sb.de

Abstract

We introduce a logic-based 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 There are 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 exam pies which are presented are taken from an operating system domain, namely from the UNIX mail domain

1 Introduction

Intelligent help systems aim at providing advanced active help to the users of complex software systems (cf. [Breuker, 1990; Thies and Berger, 1992; Norvig *et al*., 1993]), The performance of these help systems can Inconsiderably 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 users level of education and current behavior Moreover, if a planning capability is available user-specific support can be given by proposing appropriate plans which exactly is what the PHI system aims to achieve[1]

PHI (cf. the figure below) 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 m volved), *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 recuse 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 users 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.

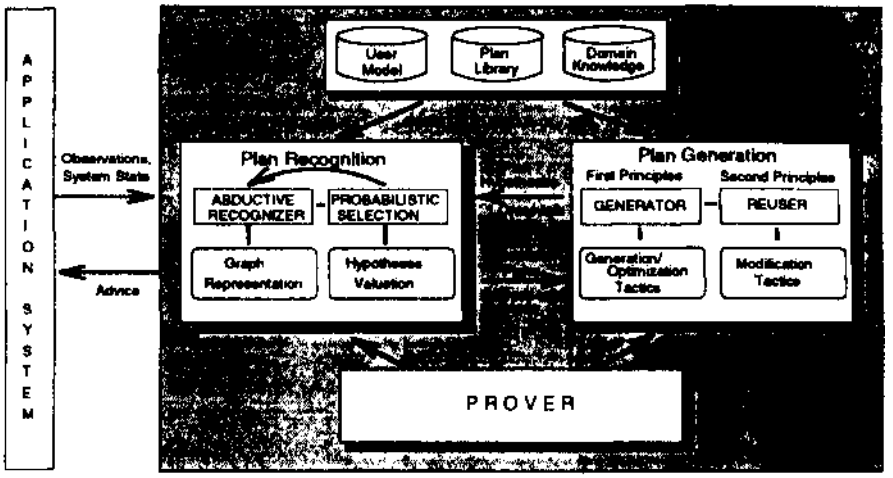
In the first cross-talk mode the plan recognizer is able to determine the most, likely plan a user follows by carrying out appropriate 'instantiations'[1] on valid plan hypotheses Thus, services like *semantic plan completion* can be offered at any time during the observation process.

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 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 follows an *abductive* principle.

The application domain, from which we present examples, is a subset of the operating system UNIX, namely its

*This work w;ts supported by the German Ministry for Research and Technology (BMFT) under contract ITW 9000 8 as part of the PHI project.



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. Finally, we conclude with some remarks in section 5.

2 The Formal Framework

Plan generation and plan recognition are carried out on a common logical basis. The logical language for plan-rung (LLP) [Biundo and Dengler, 1993], which we have developed for this purpose, combines features of Choppy Logic [Rosner and Pnueli, 1986] with the Temporal Logic for Programs [Kroger, 1987] This entails the consideration of plans as programs as has also been proposed by other authors (cf. [Green, 1969], [Bibel, 1986], [Manna and Waldinger, 1987]). 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 (e.g., conditionals) 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, is represented by the following axiom schema:

$$\forall x\,[\,[of(mb) = T \wedge df(x, mb) = F \wedge \mathsf{P}^{\,rf(x,mb)}_{\,T} \wedge EX(type(x, mb))\,] \rightarrow \bigcirc \mathsf{P}]$$

where P is a metavariable for formulas. The schema states: If the mailbox mb is open (i.e., the open-flag $of(mb)$ equals true) and message x is not yet deleted (i.e., the delete-flag $df(x, mb)$ equals false) and we execute the *type* command then formula P holds in the next state, provided the formula we obtain by replacing each ocurrence of the term $rf(x, mb)$ in P by T (i.e., the *weakest precondition* of P w.r.t. $type(x, mb)$) has held before.

The substitution instructions correspond to the effect of the *type* command. Replacing P by $rf(x, mb) = T$ we obtain the following action axiom:

$$\forall x\,[\,[\,of(mb) = T \wedge df(x, mb) = F \wedge EX(type(x, mb))\,] \rightarrow \bigcirc rf(x, mb) = T\,]$$

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 (i.e., the read-flag $rf(x, mb)$ is set to *true*).

Plans are represented by a certain class of LLP formulas. Besides basic actions (denoted using 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* $\wedge$ 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

$$of(M) = T \wedge \mathsf{Plan} \rightarrow \Diamond[displ = headers(M) \wedge \Diamond\,[rf(x, M) = T]] \qquad (1)$$

specifies the plan "Display the content of mailbox M 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:

$$EX(header(M))\ ;\ \textbf{if } df(x, M) = T \textbf{ then } EX(undelete(x, M))\ ;\ EX(type(x, M)) \qquad (2)$$

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 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:

$$of(M) = T \wedge df(x, M) = F \wedge \mathsf{Plan} \rightarrow \Diamond[displ = headers(M) \wedge \Diamond[rf(x, M) = T \wedge \Diamond[df(x, M) = T]]] \quad (3)$$

$$EX(header(M)); EX(type(x, M)); EX(delete(x, M)) \quad (4)$$

is obtained 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. [Biundo *et* al., 1992]) 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

3.1 Planning from First Principles

By using a sequent calculus for LLP (cf. [Biundo and Dengler, 1993]) 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 [Constable, 1986], [Ileisel et *al*., 1990], [Paulson, 1990]) 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 suhplans: $\mathsf{Plan} \equiv \mathsf{P}_1 ; \mathsf{P}_2$.

Let us now consider the generation of a plan for P_2 The corresponding subgoal reads

$$of(M) = T \wedge \mathsf{P}_2 \rightarrow \Diamond rf(x, M) = T$$

Usually subgoals of tins type are proven by using nonlogical axioms which describe basic actions. Thereby, 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 rf to T:

$$of(M) = T \wedge df(x, M) = F \wedge EX(type(x, M)) \rightarrow \Diamond rf(x, M) = T$$

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. Following a deductive version of the *means-ends analysis* (cf [Fikes and Nilsson, 197l], [Nilsson, 1980]) we therefore introduce an additional subplan which produces the missing precondition. Thus, P2 becomes the composition of a one-armed conditional and a subplan P4, respectively:

$$\mathsf{P}_2 \equiv [\mathbf{if}\ df(x, M) = T\ \mathbf{then}\ \mathsf{P}_3]; \mathsf{P}_4$$

The new subgoal obtained is:

$$of(M) = T \wedge \mathbf{if}\ df(x, M) = T\ \mathbf{then}\ \mathsf{P}_3 \rightarrow \Diamond[of(M) = T \wedge df(x, M) = F]$$

To properly instantiate P3 an instance of the *undelete* action axiom can be used; this tells us that the execution of *undelete*(x, M) makes $df(x, M) = F$ true in the next state, should it not have held before. In a similar way P4 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 *of (M)* = *T*—which acts as a precondition to the whole plan does survive the execution of subplan P_1. This fact is proven by regression where we generate the weakest preconditions of *of(M)* = *T* w.r.t. all basic actions occurring in P_1. In our system, planning from first principles is, like several other approaches to deductive planning (cf. [Green, 1969], [Bibel, 1986], [Manna and Waldinger, 1987]) closely related to work done on *deductive program synthesis* where *programs* are generated by proofs (cf. [Manna and Waldinger, 1980],[Heisel *et al,* 1991], [Biundo, 1992]).

3.2 Planning from Second Principles

The ability of a planner to modify a plan is considered as a valuable tool for improving the efficiency of planning by avoiding the repetition of the same planning effort because instead of generating a plan from scratch, plan reuse tries to exploit knowledge stored in previously generated plans.

The reuser first takes the current specification and searches in a *plan library* for a plan which can be reused as its solution Since we concentrate on plan modification in this paper, we suppose that the search in the plan library terminates successfully with a plan specification and describe how the reuser verifies whether the plan belonging to it provides 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 from the generation process that lead to this plan.

Let us assume, for example, that specification (3) is given to the planner in the first cross-talk mode. Planning from second principle starts and tries to reuse plan

Comparing it with specification (1) it is obvious that more preconditions are given, but even more goals are required in (3). In this case, the prover reports a failure because more goals are required in specification (3) than are achieved by the plan. The modification tactic identifies the missing subgoal $df(x, M) = T$ 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.

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

$$EX(header(Mbox)), EX(type(x, Mbox)), \mathrm{Plan}_1$$

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 (3). 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 $EX(delete(x, M))$. 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 [Biundo *et al.*, 1992; Koehler, 1992]. 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 [Kambhampati and Hendler, 1992], or in the framework of case-based reasoning, e.g., the systems SPA [Hanks and Weld, 1992] or CHEF [Hammond, 1990]. The experiments reported by some of the authors give evidence that plan reuse might indeed be more efficient than planning from scratch. How far these results generalize is studied in a complexity-theoretic analysis of plan modification vs. plan generation by Nebel and Koehler (cf. these proceedings). In contrast to practical experiences it turned out that plan modification is not uniformly as easy as planning from scratch.

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. Starting from plan hypotheses synthesized by the plan generation component and observations of user actions, an attempt is made to identify a hypothesis describing the user's pursued plan. The use of probabilistic reasoning allows us to determine *one* "best" hypothesis to offer user-specific help, e.g., by doing semantic plan completion.

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., [Appelt and Pollack 1990], [Shanahan, 1989], [Helft and Konolige, 1990],[Waern, 1992]) P is required to be a *ground instance* of an element of a set of predefined candidate explanations called *abducibles.*[1]

[1] For an introduction to abduction see, for example, [Peirce, 1931 1958] or [Fann, 1970]. An overview can be found

[Bauer, 1993] M Bauer Plan recognition under uncertainty Research Report, DFKI, 1993 to appear.

Ben-Ari et a /., 1982] M Ben-Ari, J Y Hal pern, and A Pnueli Deterministic propositional dynamic logic: finite models, complexity, and completeness Comput. System Set., 25:402 417, 1982.

[Bibel, 1986] W Bibel A deductive solution for plan generation. New Generation Comp., 4:115-132, 1986.

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

[Biundo et ai, 1992] S Biundo, D. Dengler, and J. Koehler. Deductive planning and plan reuse in a command language environment In Proc. of 10th EC AI pages 628-632, 1992

[Biundo, 1992] S. Biundo Automatische Synthe.se rekursiver Programme als Beweisverfahren Springer IFB 302, Berlin, Heidelberg, New York, 1992.

[Breuker, 1990] J Breuker. EUROHELP Dtvdoping Intelligent Help Systems. EC, Kopenhagen, 1990.

[Constable, 1986] R.L Constable Implementinq Mathematics with tht Nuprl Proof Development System Prentice-Hall, 1986

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

[Fikes and Nilsson, 1971] RE Fikcs and N.J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. AI 2:189-208, 1971

[Green, 1969] C.Green Application of theorem proving to problem solving. In Proc. of 2th JJCAI, pages 219-239, 1969.

[Hammond, 1990] K. J. Hammond. Explaining and repairing plans that fail. AI, 45:173 228, 1990.

[Hanks and Weld, 1992] S. Hanks and D S Weld Systematic adaptation for case-based planning. In Proc. of 1st AIPS, pages 96 105, Washington, D C , 1992 Morgan Kaufmann

[Heisel et al, 1990] M. Heisel, W Reif, and W. Stephan. Tactical theorem proving in program verification. In Proc. of 10th CADE Springer LNCS 449, 1990

[Heisel et «/., 1991] M. Heisel, W. Reif, and W Stephan. Formal software development in the KIV system. In Automating Software Design, R. McCartney and M.R. Lowry (eds.). AAAI Press, 1991.

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

[Kakas tt al., 1992] A C Kakas, HA 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.

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

[Koehler, 1992] J Koehler Towards a logical treatment of plan reuse. In Proc. of 1st AIPS, pages 285-286, Washington, D C , 1992. Morgan Kaufmann.

[Kroger, 1987] F. Kroger. Temporal Logic of Programs. Springer, Heidelberg, 1987.

[Kruse tt ai, 1991] R. Kruse, E. Schwecke, and J. Heinsohn. Uncertainty and Vagueness tn Knowledge Based Systems. Springer, 1991.

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

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

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

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

[Paul, 1993a] G. Paul. Approaches to abductive reasoning - an overview AI Review, 1993. to appear.

[Paul, 1993l>] G. Paul A generalized abductive principle for a modal temporal logic. Research Report, DFKI, 1993 to appear

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

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

[Pratt, 1979] V. Pratt Models of program logics. In Proc. of 20th Annual IEEE Symposium on Foundations of Computer Science, pages 115-122, 1979.

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

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

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

[Shan ah an, 1989] M. Shanahan. Prediction is deduction but explanation is abduction. In Proc. of 11th IJCAI, pages 1055-1060,1989.

[Tines and Berger, 1992] MA. Thies and F. Berger. Plan-based graphical help in object-oriented user interfaces. In Proc. of the International Workshop on Advanced Visual Interfaces', Rome, Italy, 1992.

[Waern, 1992] A. Waern. Reactive abduction. In Proc. of 10th ECAI, pages 159-163, 1992.

[Wolper, 1985] P. Wolper. The tableau method for temporal logic: an overview. Logique et Anal., 28:119—136, 1985.