# USER APPROPRIATE PLAN RECOGNITION FOR ADAPTIVE INTERFACES

Marc Hofmann and Manfred Lang

Institute for Human-Machine Communication
Technical University of Munich, D-80290 Munich, Germany
{hofmann, lang}@ei.tum.de

## ABSTRACT

Adaptive user interfaces are often based on plan recognizers, which consider just optimal action sequences to reach a goal. The algorithm we present is an approach towards user appropriate plan recognition, i.e. it stays abreast of the fact that in complex domains users often behave sub-optimal to achieve their goal because of a lack of knowledge about the domain and about its commands. Furthermore our algorithm is able to deal with goals with various ways to achieve.

## 1. INTRODUCTION

User assistance systems usually have a component, which models the user's potential goal for controlling an adaptive interface. In this paper we describe a plan recognizer to infer the user's goal regarding previously observed user actions. Our domain is a Unix file system with a standard Unix shell for entering commands. We refer to files and directories as *objects* the user may manipulate by *operators*. Operators can be interpreted as sub-plans; usually they are Unix commands or groups of Unix commands. For plan recognition we interpret a Unix plan as a vector of operators for manipulating a number of objects:

$$plan = \textbf{\textit{operator( object )}} \quad (1)$$

As mathematical basis for the plan recognition algorithm we make use of Bayesian belief networks [Pea88]. Charniak and Goldman first have used Belief networks for plan recognition [Cha92]. The main feature of our plan recognizer is its ability to exploit optimal and also sub-optimal user behaviour, i.e. the plan recognition process is user appropriate. Hence our networks differ in the topologies and probability tables from the networks of the authors mentioned. We introduced a hierarchical structuring of plan networks in four layers for an user adequate representation of plans.

## 2. METHODOLOGY

### 2.1 Requirements

To gather knowledge about typical user acting in the Unix environment and to gather training data for the plan recognizer, experimental subjects have been given various Unix tasks. Analyzing the resulting plans lead to conclusions, which lead to a number of requirements for our plan recognizer.
In our test plans, all users acted sub-optimal, i.e. they made use of commands, which are irrelevant for achieving the goal. Nevertheless these actions can be interpreted as characteristic action patterns for certain plans, which can be ascribed to a lack of knowledge of the domain and its commands. Wrong usage of commands and mistyping are also frequently made mistakes. Furthermore each user seems to have an individual approach to plan completion.
Finally our goal was to build a plan recognizer that stays abreast of imperfect acting of users and which also exploits information apart from the optimal sequence of actions for classification of plans. For an appropriate weighting of actions and sub-plans, we decided for a probabilistic mathematical fundament, namely Bayesian belief networks.

### 2.2 Basic structure

For the basic structure of our algorithm, refer to Fig. 1. The user has a certain goal he wants to achieve. Therefore he needs a solution for his problem, a sequence of actions leading to that goal. This sequence is defined as *plan*. The task of the plan recognition system is inferring the user's plan by means of previously observed actions. Therefore the plan recognizer is fitted with a plan library, a database containing all potential plans to reason about. For each

From: Proc. HCI 2001
Lawrence Erlbaum Ass., NJ

plan a *plan model* is generated as basis for the plan evaluation. As this plan model is a Bayesian belief network, we refer to the networks as *plan model networks*. With every new user action all potential plans of the plan library have to be evaluated, given the current and previous user actions as observations. For each plan hypothesis an evaluation measure EM is calculated, which reflects the belief that the observed actions are part of that plan. The plan with the maximum evaluation measure is the result of the plan recognition process.

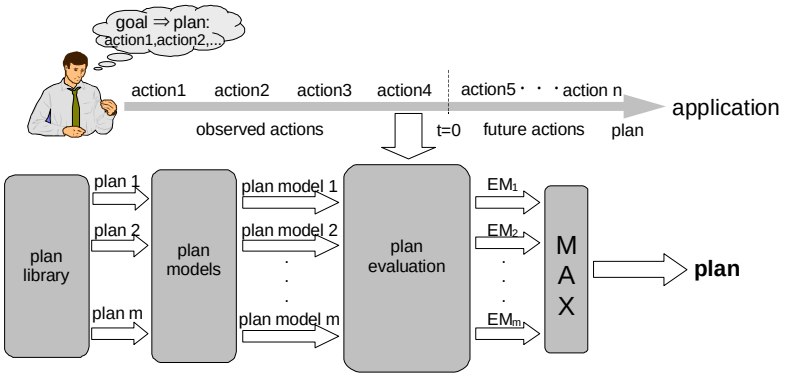


**Figure 1:** Basic structure of the plan recognition system

## 2.3 Plan model networks

When using Bayesian belief networks the main task is to find an adequate topology for the network and choosing the proper conditional probability tables for modeling a certain problem. A plan model network is structured hierarchically in four layers, which will now be described in detail.

**Plan hypothesis layer**

The plan hypothesis layer is the top layer of a plan model network. It allows direct inference of the belief that the observed user actions are part of that plan. Therefore the confidence of a plan is modeled by one discrete, Boolean state variable. At the beginning it is assigned a neutral probability distribution as a priori probability P(plan), i.e. both states are equally likely. We refer to this node as *plan node*. As we interpret a plan as the manipulation of objects, we represent each object to manipulate by a Boolean state variable, which is linked with the plan node by an arc. Moreover the object nodes are linked among each other. Fig. 2 shows the topology of the network.

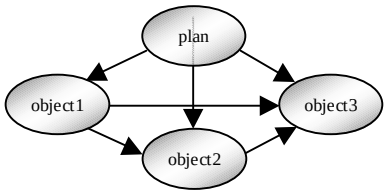


**Figure 2:** Topology of the plan hypothesis layer for a plan for manipulation of three objects

The structure of the belief network and appropriate conditional probability tables of the object nodes enable modeling the following logical AND-function:

$$plan = object1 \wedge object2 \wedge object3 \ldots \quad (2)$$

Equ. 1 reflects the fact that a plan is only completed if the user manipulated all relevant objects in the way the plan is meant for. The a posteriori probability of the plan node P( plan | object1, object2, ... ) allows inferring the completion of the plan only if the corresponding objects are manipulated correctly and completely.

**Object layer**

The object layer models how objects are manipulated. It provides one Boolean node for each operator an object has to be manipulated with. For the conditional probabilities we again have a logical AND-function of the object node with *operator nodes* as parameters. In analogy to the plan hypothesis layer the manipulation of one object is only completed if all necessary operators have been used on that object. This part of the network reflects optimal behaviour, only the essential operators to achieve a goal:

$$object = operator1 \wedge operator2 \wedge ... \quad (3)$$

Now operator nodes, which reflect sub-optimal actions, but nevertheless for that plan characteristic action patterns, are linked to the object node. We refer to these operators as *optional operators*, which may support the belief in a particular plan when observed, but the object may be also manipulated correctly and completely without using these operators. For training the conditional probabilities we gathered training data by giving a number of Unix tasks to various experimental subjects. The emerging plans are used for weighting the contribution of optional operators to plan completion. This is a task of training with complete knowledge, so the probabilities can be determined according to the frequencies of occurrence of optional operators. For weighting the contribution in that way that observing an optional operator supports the belief in a plan, only conditional probability values between 0.5 and 1 are of interest. Therefore we map the division of the number of optional operators observed in the training data ($n_{opt\ op}$) and the number of interesting manipulations of a particular object ($n_{obj}$) on the range from 0.5 to 1 for the "yes"-state. This results in the following equations:

$$P(opt\ operator = y \mid object = y) = \frac{1}{2}\left(1 + \frac{n_{opt\,op}}{n_{obj}}\right) \qquad P(opt\ operator = n \mid object = y) = \frac{1}{2}\left(1 - \frac{n_{opt\,op}}{n_{obj}}\right) \quad (4)$$

The conditional probability values reflecting the contribution of an optional operator to other plans are chosen neutral, because we treat each plan individually and independent of other plans. This fact and equ. 3 enable even very rarely observed optional operators to contribute to the belief in a plan.
Fig. 3 shows the topology of a typical object layer consisting of two operators and *n* optional operators.

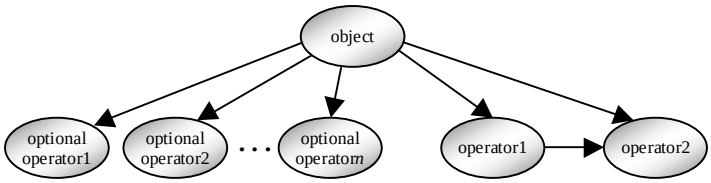


**Figure 3:** Topology of the object layer with two operators and *n* optional operators

**Operator layer**
The operator layer models the way an operator is created by a number of user actions. It provides one Boolean node for representing an operator and one Boolean for each user action. This layer also combines optimal and sub-optimal usage of actions. Fig. 4 shows the topology with *n* optional actions and a few actions for modeling optimal behaviour. As the structure shows, it's possible to model different approaches to that operator. Hence synonymous actions with different commands, different options and parameters, but with equal effect can be modeled. In Fig.4 the combination of action1 and action2 has the same effect as action3. In the case of two or more actions to create the operator, the structure and the conditional probabilities are chosen according to a logical AND.

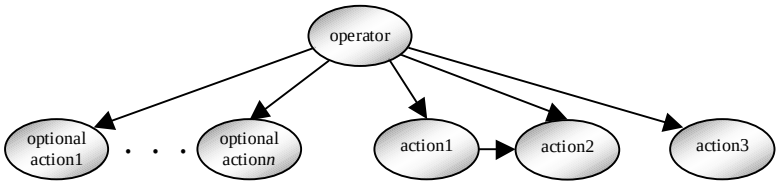


**Figure 4:** Topology of the action layer with *m* optimal actions and *n* optional actions

The conditional probabilities of the optional action nodes are chosen in analogy to the object layer.

**Action layer**
The action layer consists of nodes directly representing user actions. Each node is related to various mapping information to ensure the action is mapped on the right node as new information. For modeling an action it is decomposed according to its syntax. Fig. 5 pictures the topology of an action with the syntax pattern <command> <options> <object1> <object2>. The decomposition enables the plan recognizer to deal with mistyping, i.e. in case of a wrong syntax component we only map the observation of right components on the action layer's nodes. The arcs and conditional probabilities again are chosen according to the following logical AND-functions.

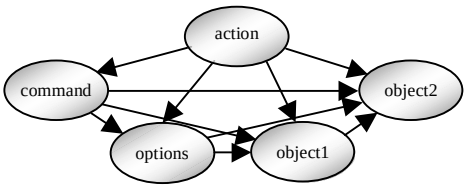


**Figure 5:** Topology of the action layer for an action with the syntax <command> <options> <object1> <object2>

The decomposition of actions does not apply to *optional action* in order to put not too much emphasis on them. Hence each optional action is represented by one single node.

## 2.4. Plan evaluation

After creating a plan model network for each plan of the plan library, the plan evaluation process can start. Fig. 6 pictures an excerpt of a plan model network, which takes a number of optional operators and optional actions into account. Below the structure user actions and their corresponding directory information are listed. Every new user action is compared with the commands, directories and objects, which are assigned to nodes. If the action matches with a node, that state variable is given the state "yes", i.e. it is instantiated and represents the observation of that user action. This information is propagated through the whole network to support the belief in that plan. User actions that are not represented by any nodes are not considered for the plan evaluation.

Commands for changing the directory ("cd") are treated in a special way. Not the command itself will be mapped on the plan model network, but the current directory. If the user changes into a certain directory, the state variable representing that directory will be instantiated. Leaving that directory results in some kind of backtracking, the instantiation of the corresponding state variable will be revoked.

Plan: **„Decompress and print all files of the directory /d1/d2"**

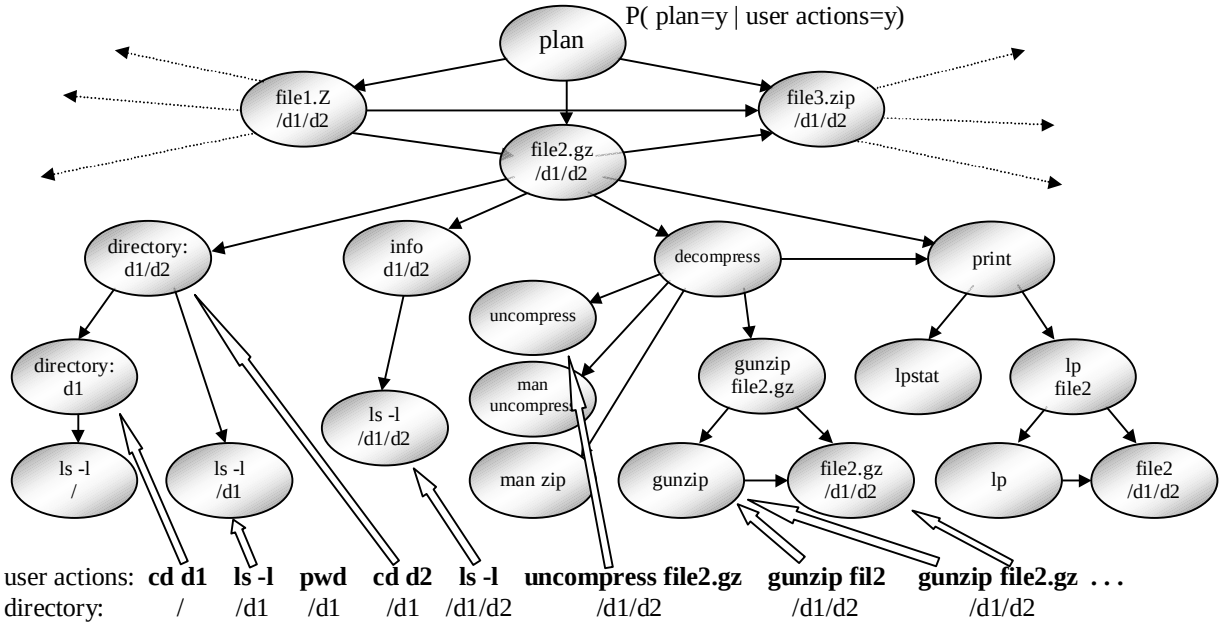


**Figure 6:** Excerpt of a plan model network and a stream of user actions

If an object has been manipulated completely by using the correct operators, the statistical dependency between the plan node and optional operators or actions for manipulating that object, is blocked, as the object node is instantiated. That means further mapping will not affect the belief in that plan. In this case we make use of the Bayesian belief network's phenomena of "d-separation" [1].

All plan hypotheses have to be treated in the same way. To compare each plan hypothesis with each other, we first calculate the belief in a plan, given the observations of previous user actions. Inferring the belief *Bel(plan)* for *k* observed actions can be done according to the following equation:

$$Bel(plan = y) = P(plan = y \mid action\,1 = y, action\,2 = y, ..., action\,k = y) \quad (5)$$

The belief has to be calculated for all plans to reason about. To be able to compare all plans, we use the belief *Bel(plan)* to determine the percentage of how complete a plan is modeled by the observed actions. As plans may consist of different numbers of objects to manipulate, we have to multiply the number of objects $n_o$ of the plan and divide it by the maximum number of objects $n_{max}$ a plan can have. The result is the following equation for the evaluation measure *EM*:

$$EM = (Bel(plan = y) - 0.5)\frac{n_o}{n_{\max}} = (P(plan = y \mid action\,1 = y, action\,2 = y, ..., action\,k = y) - 0.5)\frac{n_o}{n_{\max}} \quad (6)$$

The evaluation measures of all plans have to be calculated. The plan with the maximal evaluation measure is the plan to decide for.

## 3. RESULTS

Evaluating a plan recognizer quantitatively is always a crucial task, because the recognition rate heavily depends on the degree of completion of the plan. As the main feature of our algorithm is the user appropriate plan evaluation, we tested the algorithm as the main component of a user assistance system [Hof01]. A number of experimental subjects with different Unix-skills have been given a number of tasks with the assistance system offering partial task completion on the basis of the plan recognizer's output. The plan library consisted of 20 plans. Figure 7 proves the acceptance of the whole assistance system with the plan recognizer as the central component. The target group, users with little or medium Unix experience judged the assistance system it to be helpful. It has been expected that Unix experts didn't accept the assistance as the system hasn't been created to cope with their needs.

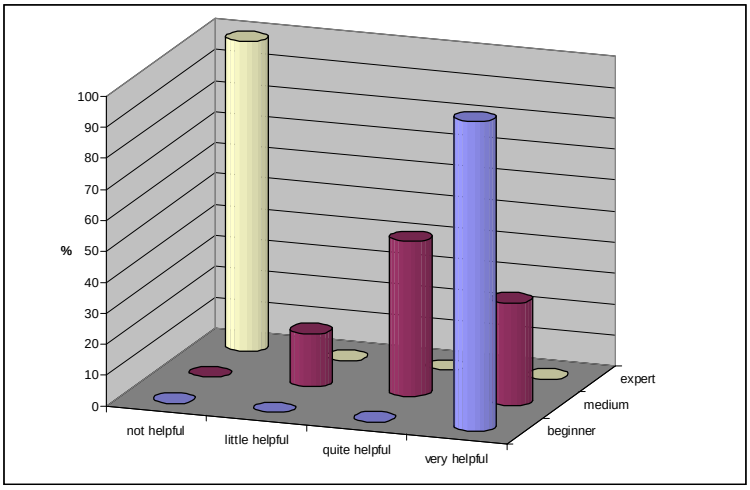


**Figure7:** Result of an investigation on the acceptance of the plan recognizer based assistance system

## 4. CONCLUSIONS

Our plan recognizer proved to work well, especially its main feature, the ability to make use of optimal as well as sub-optimal user acting proved to be the key for plan recognition for user assistance systems. Future work will be left to the automatic and dynamic generation of the plan library.

## REFERENCES

[Pea88] Pearl, J.: "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann, California, 1988

[Cha92] Charniak, E. and Goldman, R.B.: "A Bayesian Model of Plan Recognition", Artificial Intelligence, 64(1), 1992, 53-79

[Hof01] Hofmann, M. and Lang, M.: "A Dialog Model for Offering Task Completion for complex Domains", Poster Proceedings HCII 2001 (New Orleans, Lousiana, USA), (this conference)