User-Tailored Plan Generation

Detlef Küpper, Alfred Kobsa

GMD FIT, German Nat'l Research Center for Information Technology, D-53754 St. Augustin, Germany

Abstract. The output of advice-giving systems can be regarded as plans to be executed by the user. Such plans are fairly useless if the user is not capable of executing some of the involved plan steps, or if he does not know them. We propose a two-phase process of user-tailored plan generation and plan presentation to produce advice that enables a user to reach his goals. This paper reports the first phase, the generation of a plan under the constraints of the user's capabilities. The capabilities are represented as a hierarchy of plan concepts. System assumptions about user capabilities form a part of the user model, but are separate from assumptions about the user's knowledge, goals etc. With this representation, we can re-use the techniques for collecting assumptions about the user's conceptual knowledge for inferring his capabilities as well. We show an example of plan generation for users with different capabilities.

1 Introduction

Adapting system responses to users is a central area of user modeling. However, the question of whether or not such responses really enable a user to reach his goals did not receive much attention so far. For advice-giving systems like online help and assistance systems this is however a crucial issue. We take the view that advice for a user is a proposed plan that the user should execute. The user must understand the system's advice and must be capable of executing the plan in order to reach the intended goals. We propose a two-phase process for creating such advice. The first phase generates a plan according to the user's capabilities. This plan is then the starting point for the second phase which creates a presentation of that plan taking the user's knowledge into account. Our approach to the problem of user-tailored plan generation will be illustrated within the context of a help system for a typical computerized office environment. There we assume a heterogeneous network of computers with different operating systems and users with various capabilities.

This paper describes the first phase in generating a user-tailored plan. We first describe the extent to which AI research on plan generation considers user-tailored planning. Then we define the notion of capabilities as we use it in this approach and discuss how a user model may represent such capabilities. We employ the user modeling shell system BGP-MS[1] (Kobsa and Pohl, 1995) to represent user models and demonstrate example user models for five different users. The partial order planner UCPOP (Penberthy and Weld, 1992) then generates plans for an example goal, taking the capabilities of these users into account.

---

[1] We use a slightly extended version of release 2.2. It provides means for the definition of new modalities and allows stereotypes for all defined modalities (Pohl and Höhle, 1997).

2 Planning

In the field of user modeling, plan processing mainly deals with plan libraries for plan recognition (e.g. Carberry, 1990, Kautz, 1991, Bauer, 1996). Only very few help systems generate plans (e.g., Breuker, 1990), however not user-specific ones. Plan libraries are assumed to contain all meaningful plans in an application domain. This approach, however, does not scale up very well when a large number of possible plans must be considered, or when plans have many variants. Unfortunately the number of potential plans increases exponentially with the number of tools that an agent may use.

Plan libraries also presume that the domain remains fairly static so that changes in the plan libraries do not occur very often. We therefore prefer plan generation (planning) over plan libraries, even though it is computationally expensive (Bylander, 1991, Erol et al., 1995). A static plan library approach even seems meaningless for user-tailored planning since it must not only contain plans for every possible goal but also plan variants for each possible combination of relevant user capabilities.

For an assessment of whether or not AI planning might extend to plan generation for a user, we take a short look at its main ideas. Planning means searching for a sequence of plan steps - the *solution* or the *plan* - that achieves the *goal* by executing the plan steps in a required order provided that the execution begins in a *start situation*. A plan step is an instance of a plan operator; it is defined by its precondition and its effect. Planning systems require goals, situations, preconditions and effects to be described by a restricted first order logic expression - usually a conjunction of literals. Even though today's view of plan *generation* is quite different[2], the following description may still give a rough idea of what happens when a plan is being *executed*. Executing a plan step in a situation *s* changes the situation by adding all literals of the step's effect and deleting all literals of *s* that contradict them. A plan step may only be executed in a situation that fulfills its precondition. We may imagine that reaching the goal by executing a plan means changing the start situation step by step until we finish with the last step in an end situation that fulfills the goal. A correct plan guarantees the precondition of each step to hold just before its execution.

Clearly a traditional planning process presumes the acting agent to be capable of executing all plan steps - otherwise the execution of the plan would fail. The fixed set of operators that the planner uses for generating plans defines the *activity potential* of the agent. For an artificial agent like a robot, this set is defined by the functional interface of the agent. We may likewise think of modeling the activity potential of users by sets of plan operators.

An example of one such operator is *open a file with text processor of type X* (open-file-tpx). We describe it with the syntax of the planner UCPOP:

```
(:operator open-file-tpx
 :parameters   (?app ?f)
 :precondition (and (tpx-file ?f) (local-fs-object ?f)
                    (tpx ?app) (launched ?app))
 :effect       (active ?app ?f))
```

----

[2] See e.g. (Russell and Norvig, 1995) for an overview of planning, (McDermott and Hendler, 1995) for an overview of the evolution of the view of planning, and (Weld, 1994) for extensions of precondition and effect descriptions and their consequences.

The meaning of this expression is that the agent may execute the operator if there is a local file (local-fs-object) that is appropriate for text processor X (tpx-file) and a launched text processor of type X (launched ∧ tpx). After the execution of this operator, this file is the active file of that text processor. Names of variables are marked by the prefix "?".

This example directly leads to the question of what is an adequate level of abstraction for this set of all operators. Here we have to distinguish between plan presentation and plan generation. The central task of plan presentation is to choose a level of abstraction that does not exhibit redundant and possibly misleading details, but on the other hand provides sufficient information enabling the user to recognize the intended plan (see, e.g. Young, 1996). In the field of plan generation, hierarchical task network (HTN) planning (Sacerdoti, 1977, Erol et al., 1994) deal with abstraction. They exploit several levels of abstraction for improving the efficiency of the planning process. At the end of the day, however, a plan must include all details because otherwise correctness cannot be guaranteed. This means that plan generation has to use the most detailed operators. But what *most detailed operators* means differs between the users, due to the different level of system's knowledge about users (see chapter 5 for an example).

3 Users' Abilities and Knowledge

The user's activity potential in a domain is mainly determined by his abilities and his authorization. The ability to climb stairs or to hit a special key combination on a computer keyboard are examples of the former. While normal users are able to perform such actions, some handicaps may prevent people from carrying them out. Examples of actions that usually depend on authorization are: *reading mail* (the user needs, e.g., an account), *open a ftp connection* (e.g., the user needs to pass a firewall), *printing on an expensive printer* (e.g., the user needs an account).

For plan generation, it makes no difference *why* a user is assumed to be capable of some action. We may define plan operators for all kinds of actions a user is capable of. This leads to a definition of the term *activity potential* of a user for the purpose of our approach:

> The *activity potential* of a user is the set of plan operators that the user is in principle able and authorized to execute.

The proviso *in principle* has been added since we also consider plan operators that users currently cannot execute due to lack of knowledge or due to unfulfilled preconditions. The above definition may be extended to arbitrary agents by replacing 'user' with 'agent'.

Some terminological remarks: Several researchers distinguish between the user's knowledge (which must be true) and his beliefs (which may be wrong). Since we do not need an explicit representation of this distinction we employ the terms interchangeably. Also, we will use the shorter term *capabilities* as a synonym for activity potential in the remainder of this paper. This should not be confused with *inferential capabilities* of users, which are investigated in text planning (e.g. Horacek, 1997, Zukerman and McConachy, 1993).

It is important to note that the user's capabilities should not be mixed up with the user's beliefs about plans. In our system, we strictly separate (system assumptions about) the user's capabilities from (system assumptions about) his beliefs - especially his beliefs about plan operators. Most real-world actions require both. For instance, users may not know how to reboot a DOS-PC, even if they are physically capable of hitting the famous key combination. On the other hand, some

users, such as those who are handicapped, may not be able to hit these keys even when they have the required knowledge.

This separation of users' beliefs from their capabilities is not very common in AI systems. This is not surprising since a distinction is not really necessary in their application domains. Planning systems, for instance, usually generate plans for robots, and not for people. The capabilities of robots are determined by their functional interface. Therefore, the commands they accept as an input define exactly what they can do. A separation of beliefs (whatever this might be for a robot) from capabilities makes no sense when planning for robots. The same holds true for plan recognition systems that recognize user plans based on their actions. Such systems only see those actions that the user both knows and is able to perform.

In contrast, advice-giving systems that want to enable users to reach their goals do need to separate capabilities and knowledge. Lack of knowledge is far less critical than lack of capabilities since it can be overcome by instruction. This leads to the two-phase process for advice generation mentioned above. The first phase generates a plan from the operators of the user's activity potential, disregarding any assumptions about the user's beliefs. The presentation phase then generates an explanation of those plan steps that are unknown to the user. If we were to restrict plan generation to those operators only that are both in the user's activity potential *and* known by him, we would find less plans and specifically lose the opportunity of explaining plan steps to the user which he is capable to perform. The benefit of such help systems would be rather limited.

Although this paper does not deal with plan presentation, we should note that this two-phase process is not unidirectional. For instance, if a candidate plan generated in the first phase requires lengthy explanations in the presentation phase because many plan steps are unknown to the user, the presentation component may request the generation of an alternative plan.

Our model presumes easily changeable beliefs of the user, while his capabilities are assumed to be constant. We assume that the capabilities of a user do not change during the interaction with the system, nor during the execution of an advised plan. However this does not mean that the system's assumptions about the user's capabilities cannot change. In our system, updating the *assumptions* about the user's capabilities is done by the same techniques as updating assumptions about the user's beliefs.

4 Representation of concepts and plans

The user's beliefs about domain concepts form a significant part of his knowledge. Many user models represent this as an abstraction hierarchy of concepts with a terminological knowledge representation system (e.g., Sleeman, 1985, McCoy, 1989, Sarner and Carberry, 1992, Kobsa et al., 1994). In our system, we use SB-ONE (Kobsa, 1991) for this purpose, which forms part of the user modeling shell BGP-MS. SB-ONE fits loosely into the KL-ONE-paradigm (Brachman and Schmolze, 1985). With this approach a concept may be defined by its superconcepts and its attribute descriptions, i.e. its relations to other concepts (these relations are called *roles*). We adopt this approach for also representing beliefs about plan operators in conceptual abstraction hierarchies since we want to exploit the rich research experience in this area (e.g., for explaining concepts to the users and gathering conceptual knowledge of the user) and also take advantage of the inferential services of terminological representation systems (e.g., inferences over subsumption and disjointness relationships).

The same arguments hold true for assumptions about the user's capabilities. Therefore we represent capabilities and beliefs in a uniform way. We call the terminological representation of a plan operator a *plan concept.*

Some representations of plans in abstraction hierarchies disregard preconditions and effects (Kautz, 1991, Weida, and Litman, 1992). While this is often sufficient for the purposes of plan recognition, it is insufficient for the generation and explanation of plans. This view is also found in (Devanbu and Litman, 1996). We extend their approach to provide plan operators with variables. This leads to a different view of what preconditions and effects are. Devanbu and Litman model them as situations (sets of predicates that are expected to be true). We model preconditions as objects that are needed before, and effects as objects that are available after the execution of plan operators. We represent the relation of these objects to their plan concepts by roles with a number restriction of exactly one. Properties of the objects are represented by value restrictions on these roles, and relations between the objects are expressed by role-value-maps. Since plan operators change properties, we cannot use identical objects for preconditions and effects. Therefore we distinguish between object descriptions for preconditions and for effects.
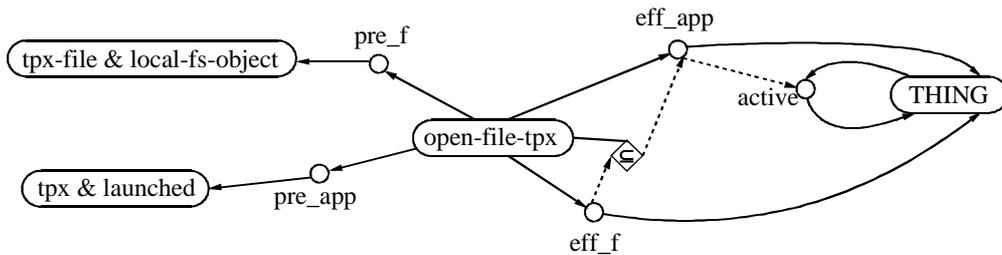

Figure 1. Terminological representation of the operator open-file-tpx in SB-ONE.

Figure 1 shows the terminological representation of the operator open-file-tpx in SB-ONE. The roles pre_app and pre_f specify that this plan concept needs two objects as preconditions: one of type tpx that is launched (a text processor of type X), and one that satisfies the conditions of a local-fs-object and a tpx-file (a local file that is appropriate for text processor X). The effects of the plan concept are expressed by roles eff_app and eff_f, and a role-value-map eff_f $\subseteq$ eff_app $\circ$ active. They specify that the execution of the plan does not introduce additional type constraints on the objects since the chosen type is the most general concept THING. A relation *active* between the two objects is however introduced which expresses that the active document of the application will be identical to the document that becomes opened by open-file-tpx.

5 Structure of a User Model with Capabilities

In this section we present a representation for all ingredients of our model, namely knowledge about plan operators and their abstraction hierarchies, system assumptions about users' capabilities and beliefs, and stereotypes that contain default assumptions about certain user subgroups. The

partition approach (Cohen, 1978), which is supported by BGP-MS, is an elegant means for distinguishing between capabilities and beliefs in the user model representation.

We will use an example that includes the following agents:

Oscar   a member of staff, a typical user of office programs
Trixie   a guest who is a typical Unix user
Nick   a skilled student
Theo   a less skilled student
Sue   a person that is not well known to the system

Figure 2 shows the structure of all system assumptions about the beliefs of the users, of belief stereotypes and the system's beliefs about the domain. Rectangles denote partitions. The labels outside of partitions are partition names. All partition names in this figure end with "B", which marks them as belief partitions. For instance, the partition SBSueB contains the system's beliefs (i.e., assumptions) about Sue's beliefs. Partitions that are not leaves are stereotype partitions that are labeled with their names and the suffix "B". The labels within rectangles are abbreviations of plan concept names that belong to the respective partition (for reasons of simplicity we only show plan concepts). The arrows between partitions express an inheritance relation between two partitions (in the opposite direction of the arrow). Inherited concepts are not shown, i.e., although SBOscarB is empty the system assumes Oscar to know the concepts app-print, user-get, launch and open-file-tpx. The abbreviations of plan concept names mean the following: app-print (print the file that is active in an application), user-get (user gets an object), launch (an
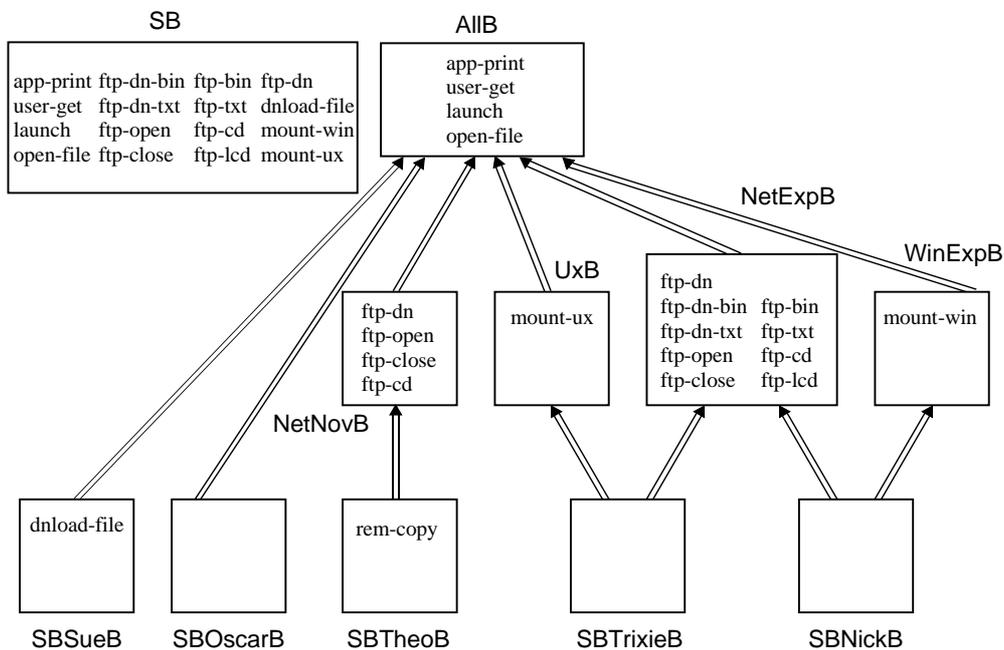

Figure 2. Structure of the *belief part* of the example user model.

application), open-file-tpx (open a file with text processor of type X), ftp-download-file-bin (download file with ftp in binary mode), ftp-download-file-txt (the same in text mode), ftp-open (open ftp connection), ftp-close, ftp-bin (change ftp-mode to binary), ftp-txt (change ftp-mode to text), ftp-cd (change remote directory of ftp), ftp-lcd (change local directory of ftp), ftp-download-file (download file with ftp disregarding file type), download-file (abstract download of a file), mount-win (mounting a directory with windows), mount-ux (the same with Unix), remote-copy (copy from or to a different host).

The example shows the assumed beliefs of the mentioned agents and several stereotypes for beliefs about different aspects of computer know-how in the areas of operating systems and networking. Partition SBTheoB contains beliefs that the system does not share. Agent Theo is assumed to believe in the existence of plan concept *remote-copy* that allows arbitrary copy operations from other computers. The system does not hold this belief.

The next figure shows the system's assumptions about the agents' capabilities. All labels of capability partitions have the suffix "C". As we separate the assumptions about capabilities of users from their beliefs, we also need a separate set of stereotypes for capabilities.

While the belief stereotypes of our example represent assumptions about typical beliefs of certain user subgroups, the corresponding capability stereotypes represent typical permissions in the example domain. Staff members have unlimited permissions, guest staff have no access to the Internet (via ftp), and students have no access to other computers of the local network (via mount).
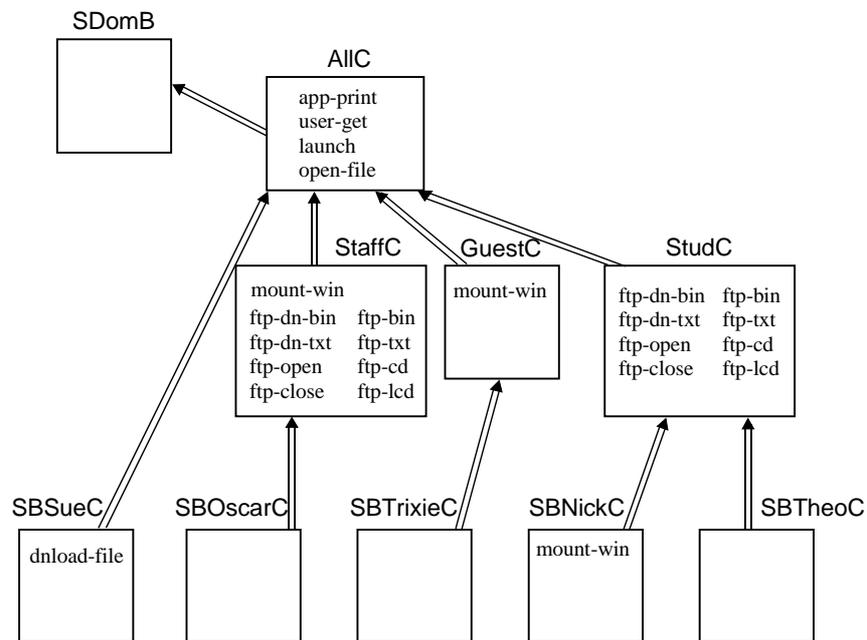

Figure 3. Structure of the *capability part* of the example user model.

SDomB is a special partition that contains the system's beliefs[3] about domain concepts (i.e. any concept that is not a plan concept). We do not show any contents of SDomB because we restrict the figures to plan concepts, as mentioned above.

6 Planning for Users - an Example

This chapter explains how the two different models of users' capabilities and beliefs influence the generation of a user-tailored plan. The example goal is:

> The user wants to have a printout of the file xy.doc that is located on the remote host konstanz. More precisely, he wants to have a new paper document that shows the contents of that file.

Note that this rather simple goal requires two different tools: a text processor and a ftp-program (or some mounting-tool). This is beyond the scope of usual help systems and even help assistants.

Let constants F-KN-xy-doc denote the file in question, and USER the user. Then we may write this goal in Predicate Logic:

$$\exists \, info \quad \text{fso-contents(F-KN-xy-doc, } info) \, \wedge \, \text{paperdoc(Dx)} \, \wedge \, \text{pd-contents(Dx, } info) \, \wedge \\ \text{has(USER, Dx)}$$

The same expression according to the UCPOP syntax:

```
(exists (fso-contents F-KN-xy-doc ?info)
        (and (paperdoc Dx) (pd-contents Dx ?info)
             (has USER Dx)))
```

Assume further that constant *Dx* does not denote any object of the domain before the execution of the plan, i.e., free (Dx) is true while p (Dx) is false for any other predicate p[4]. After the execution, the goal expression requires Dx to denote a paper document, i.e., paperdoc (Dx) is true.

With the nomenclature of the last chapter, the user must use a text processor of type X to print the file. This requires the file to be located in the local file system of the user's computer. Depending on the capabilities of the different users and on whether or not konstanz exports the relevant directory, the goal may be achieved by downloading the file (via ftp, or in an arbitrary way for Sue) or by mounting an appropriate directory of host konstanz. The results of the plan generation are shown in Table 1.

Except for Trixie, all users can in principle reach the goal in both situations. For plans in italics, at least one plan step is unknown to the user according to the user model. In this case, the user needs an explanation of the steps in question or the planner has to generate an alternative plan. Note that this is the fact even for Theo who knows ftp-download but not the required and more special ftp-download-bin. This must be distinguished from Sue's plan. The system cannot go into details (i.e., refine the plan) because it does not know how Sue can perform the download.

Note that in our example it is impossible to generate a common plan that would work for all users.

----

[3] SDomB is a direct super partition of SB. For clarity, we omitted it in figure 2.

[4] We omit a full presentation of the extensive representation of the start situation, which contains all facts in the situation (like that F-KN-xy-doc is a tpx-file, is located in directory /home/otto/docs on host konstanz, etc.). A complete description is available from *http://zeus.gmd.de/~kuepper/UM99/*.

Table 1. The results of the plan generation.

User	Directory is exported	Directory is *not* exported
Oscar	*mount*	*ftp*
Trixie	*mount*	fails
Nick	mount	ftp
Theo	*ftp*	*ftp*
Sue	download	download

where	stands for the plan
mount	launch (text processor), mount-win (appropriate directory), open-file-tpx, app-print, user-get
ftp	launch (text processor), launch (ftp program), ftp-open-connection (to computer konstanz), ftp-lcd & ftp-cd (appropriate directories), ftp-binary-mode, ftp-download-file-bin, open-file-tpx, app-print, user-get
download	launch (text processor), download-file, open-file-tpx, app-print, user-get

7 Conclusion

This paper introduced plan generation for advice-giving that considers users' capabilities and knowledge. We used a partial order planner that obtains its working set of plan operators from a user model. Modeling users' plan execution capabilities in a similar way as users' beliefs allows one to employ standard user model acquisition techniques. We demonstrated the use of stereotypes.

The strict separation of beliefs and capabilities allows for the generation of plans that include steps that are unknown to the user, rather than restricting plan generation by the user's knowledge. This separation grants advice-giving systems the option to explain unknown plan steps to the user, and leads to the next research question of generating a system response that enables a user to reach his goals in the domain. It seems promising to extend the ideas of M. Young (Young, 1996) for serving the needs of various users to fit into the approach of this work.

References

Bauer, M. (1996). Acquisition of user preferences for plan recognition. In *Proceedings of 5th International Conference on User Modeling*, Kailua-Kona, HI, 105-112.

Brachman, R., and Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science 9*:171-216.

Breuker, J., ed. (1990). *EUROHELP: Developing Intelligent Help Systems*. EC, Kopenhagen.

Bylander, T. (1991). Complexity results for planning. In *Proceedings of 12th International Joint Conference on Artificial Intelligence*, Sidney, Australia, 274-279.

Carberry, S. (1990). *Plan Recognition in Natural Language Dialogue*. MIT Press, Cambridge, MA.

Cohen, P. (1978). On knowing what to say: planning speech acts. Tech.Report 118, Department of Computer Science, University of Toronto, Canada.

Devanbu, P., and Litman, D. (1996). Taxonomic plan reasoning. *Artificial Intelligence 84*:1-35.

Erol, K., Hendler, J., and Nau, D. (1994). Semantics for hierarchical task-network planning. Tech.Report CS-TR-3239, Computer Science Dept., University of Maryland.

Erol, K., Nau, D., and Subrahmanian, V. (1995). Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence 76*:75-88.

Horacek, H. (1997). A model for adapting explanations to the user's likely inferences. *User Modeling and User-Adapted Interaction 7*:1-55.

Kautz, H. (1991). A formal theory of plan recognition and its implementation. In *Reasoning about Plans,* Allen, Kautz, Pelavin, and Tenenberg, eds., Kaufmann, San Mateo, CA. 69-126.

Kobsa, A. (1991). Utilizing knowledge: the components of the SB-ONE knowledge representation workbench. In *Principles of Semantic Networks*, Sowa, J., ed., Kaufmann, San Mateo, CA. 457-486.

Kobsa, A., Müller, D., and Nill, A. (1994). KN-AHS: an adaptive hypertext client of the user modeling system BGP-MS. *Proceedings of the 4th International Conference on User Modeling*, Hyannis, MA, 99-105.

Kobsa, A., and Pohl, W. (1995). The BGP-MS user modeling system. *User Modeling and User-Adapted Interaction 4*:59-106.

McCoy, K. (1989). Generating context-sensitive responses to object-related misconceptions. *Artificial Intelligence 41*:157-195.

McDermott, D., and Hendler, J. (1995). Planning: what it is, what it could be. *Artificial Intelligence 76*:1-16.

Penberthy, J., and Weld, D. (1992). UCPOP: A sound, complete, partial order planner for ADL. In *Principles of Knowledge Representation and Reasoning - Proceedings of the 3rd International Conference KR'92*, Cambridge, MA, 103-114.

Pohl, W., and Höhle, J. (1997). Mechanisms for flexible representation and use of knowledge in user modeling shell systems. In *User Modeling: Proceedings of the 6th International Conference, UM97*, Chia Laguna, Italy, 403-414.

Russell, S., and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle, River, NJ.

Sacerdoti, E. (1977). *A Structure for Plans and Behavior*. Elsevier/North-Holland, Amsterdam, London, New York.

Sarner, M., and Carberry, S. (1992). Generating tailored definitions using a multifaceted user model. *User Modeling and User-Adapted Interaction 2*:181-210.

Sleeman, D. (1985). UMFE: A user modelling front-end subsystem. *International Journal of Man-Machine Studies 23*:71-88.

Weida, R., and Litman, D. (1992). Terminological reasoning with constraint networks and an application to plan recognition. In *Principles of Knowledge Representation and Reasoning - Proceedings of the 3rd International Conference KR'92*, Cambridge, MA, 282-293.

Weld, D. (1994). An introduction to least commitment planning. *AI Magazine 4*:27-61.

Young, M. (1996). Using plan reasoning in the generation of plan descriptions. In *Proceedings of 13th National Conference on Artificial Intelligence*, Portland, OR, 1075-1080.

Zukerman, I., and McConachy, R. (1993). Generating concise discourse that addresses a user's inferences. In *Proceedings of 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1202-1207.