March 1987

Implicit Acquisition of User Models in Cooperative Advisory Systems

Robert Kass
University of Pennsylvania

Implicit Acquisition of User Models in Cooperative Advisory Systems

Abstract

User modelling systems to date have relied heavily on user models that were hand crafted for use in a particular situation. Recently, attention has focused on the feasibility of *general user models*, models that can be transferred from one situation to another with little or no modification. Such a general user model could be implemented as a modular component easily integrated into diverse systems. This paper addresses one class of general user models, those general with respect to the underlying domain of the application. In particular, a domain independent user modelling module for cooperative advisory systems is discussed.

A major problem in building user models is the difficulty of acquiring information about the user. Traditional approaches have relied heavily on information that is pre-encoded by the system designer. For a user model to be domain independent, acquisition of knowledge will have to be done *implicitly*, i.e., knowledge about the user must be acquired during his interaction with the system.

The research proposed in this paper focuses on domain independent implicit user model acquisition techniques for cooperative advisory systems. These techniques have been formalized as a set of model acquisition rules that will serve as the basis for the implementation of the model acquisition portion of a general user modelling module. The acquisition rules have been developed by studying a large number of conversations between advice-seekers and an expert. The rules presented are capable of supporting most of the modelling requirements of the expert in these conversations. Future work includes implementing these acquisition rules in a general user modelling module to test their effectiveness and domain independence.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-87-05.

IMPLICIT ACQUISTION OF USER MODELS IN COOPERATIVE ADVISORY SYSTEMS

Robert Kass

MS-CIS-87-05

LINC LAB 49

Department Of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

March 1987

Acknowledgements: This research was supported in part by DARPA grants NOOO14-85-K-0018, NSF-CER grant MCS-8219196 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027 and by a grant from the Digital Equipment Corporation.

Implicit Acquisition of User Models in Cooperative Advisory Systems

A Dissertation Proposal

Robert Kass[1]
Department of Computer Science/D2
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104

March 21, 1987

[1]This work was supported by a grant from the Digital Equipment Corporation

Abstract

User modelling systems to date have relied heavily on user models that were hand crafted for use in a particular situation. Recently, attention has focused on the feasibility of *general user models*, models that can be transferred from one situation to another with little or no modification. Such a general user model could be implemented as a modular component easily integrated into diverse systems. This paper addresses one class of general user models, those general with respect to the underlying domain of the application. In particular, a domain independent user modelling module for cooperative advisory systems is discussed.

A major problem in building user models is the difficulty of acquiring information about the user. Traditional approaches have relied heavily on information that is pre-encoded by the system designer. For a user model to be domain independent, acquisition of knowledge will have to be done *implicitly*, i.e., knowledge about the user must be acquired during his interaction with the system.

The research proposed in this paper focuses on domain independent implicit user model acquisition techniques for cooperative advisory systems. These techniques have been formalized as a set of model acquisition rules that will serve as the basis for the implementation of the model acquisition portion of a general user modelling module. The acquisition rules have been developed by studying a large number of conversations between advice-seekers and an expert. The rules presented are capable of supporting most of the modelling requirements of the expert in these conversations. Future work includes implementing these acquisition rules in a general user modelling module to test their effectiveness and domain independence.

Contents

List of Figures

Chapter 1

Introduction

Computer systems are reaching a point where they have begun to take on responsibilities that normally require significant human intelligence. Many expert systems encode the knowledge of humans recognized as experts in a field and are able to solve problems that normally required these human experts. In other areas, computer systems are being used to provide intelligent tutoring assistance, intelligent on-line help for computer systems and cooperative responses to data base queries that require knowledge and reasoning abilities.

As computer systems begin to play roles traditionally been associated with significant human intelligence, people will expect these systems to exhibit other aspects of intelligent behavior. In particular, such systems will be expected to *interact* with users in an intelligent manner. One aspect of intelligent interaction is that an individual can recognize the level of knowledge conversational partners have, and use this information to guide his or her behavior. For computer systems to interact with people intelligently, they will need to maintain information about the individuals with whom they interact.

Acquiring, maintaining and using information about the users who interact with a computer system is the focus of *user modelling.* User models have been incorporated in several systems to improve the level of interaction and the performance of the system. With this demonstrated success, user models have been recognized as important for supporting interaction between system and user.

User modelling has many problems, however. One of the greatest is the difficulty in building a user model. Existing user modelling systems have needed a significant amount of knowledge about users to be hand-coded by the system designers. This hand-coding is time consuming, sometimes requiring more effort than the encoding of the underlying domain knowledge. At the same time, this information is very specific to a particular system. Thus when a new system is developed for a different application, the information previously encoded is of little or no use in building the new user model.

Other user modelling problems include: determining what information should be represented in the model, how to represent the information, how to maintain information in the model, how the user model should be used, and many others. These problems are faced by each system designer who wishes to incorporate a user model into a system. The commonality of these problems has lead researchers to look at the possibility of *general user models*, which have a defined set of features and which can be used in a variety of applications.

The research discussed in this paper is concerned with general user modelling. A major problem with general user modelling is the difficulty of acquiring user models. The main point of this paper is that substantial amounts of information about the user can be acquired in a domain independent way. This makes possible the development of a general user modelling module that can be incorporated into a variety of systems with minimal effort. The following paragraphs outline the rest of this paper.

Chapter 2 focuses on user models and general user modelling. One goal of this chapter is to provide a solid characterization of what a user model is—a task more difficult than one might think. A second goal of the chapter is to discuss the notion of general user modelling. Three aspects of generality can be involved in a general user model: generality with respect to the range of possible users to be modelled, generality regarding the form of interaction between system and user, and generality with respect to the domain of the underlying application. With the terms "user model" and "general user model" both well defined, the long range goal of research in general user modelling is discussed. This goal can be labelled "the general user modelling dream."

Chapter 3 focuses on one problem of user modelling crucial to general user modelling: the model acquisition problem. Most existing user modelling systems have relied primarily on *explicit* user model acquisition. This means that information is placed directly into the user model, sometimes by the user, but more often by the system designers. For a user model to be truly general, it must be easily transferable from one system to another. If this transferal requires a large amount of re-coding of the knowledge in the user model, the goal of user model generality cannot be achieved.

Developing a general user model thus depends on acquisition techniques that are not explicit. Chapter 3 goes on to discuss *implicit* user model acquisition. Implicit user model acquisition requires the user model to be built as the system interacts with the user. This means the user modelling component must be able to "eavesdrop" on the interaction between the system and the user, and extract as much information about the user as possible. If the implicit model acquisition techniques employed by the user model are independent of the type of the interaction, domain and type of user, a general user modelling module can be achieved.

General, implicit user model acquisition is very difficult. In chapter 4 a class of systems called *cooperative advisory systems* are described. By focusing only on these systems, the range of interaction is restricted, enabling the development of user model acquisition techniques that are domain general. Cooperative advisory systems were chosen because they form an interesting and useful, yet difficult class of systems for user modelling.

Chapter 5 contains the major contributions of this research. Transcripts of a large number of conversations in an advice-giving setting were studied. These transcripts provide examples of interactions between advice-seekers and an expert who seeks to give cooperative help, providing a paradigm for how a cooperative advisory system might be expected to behave. From these transcripts, a collection of user model acquisition rules have been developed which, taken together, can generate a user model capable of supporting the behavior exhibited by the expert in the transcripts.

These acquisition rules can be loosely organized into three categories: communication rules, model-based rules and human behavior rules. Communication rules focus on statements made by the user or the system, and include several rules inspired by Grice's *cooperative principle* [Grice 75]. The model-based rules are triggered by certain configurations of information in the user and domain models. The human behavior rules depend on certain aspects of human behavior that seem to be universal. The acquisition rules are not infallible: exceptions to each can be found. Thus these rules should not be thought of as absolute inferences, but rather as reasonable rules that can be used to infer beliefs of the user. These inferences, however, may need to be retracted on occasion.

Chapter 6 describes my future research plans. The main goal of this research is to implement and test the implicit user model acquisition rules. In doing so, I plan an implementation that can be augmented to form a complete general user modelling module. Thus, in chapter 6, an architecture for such a module is sketched, and the portions that will be implemented are described.

Chapter 2

General User Models

This chapter addresses the topic of *general user modelling.* Of necessity the process will be somewhat iterative. An initial definition for "user model" is proposed that can serve as a basis for discussing the requirements of a user model, and the various generality criteria for user modelling. This discussion will identify weaknesses in the original definition, so a revised definition is proposed. Finally, the long range goals for general user modelling are briefly discussed.

2.1 An Initial Definition of User Model

Despite its apparent simplicity, the term "user model" is not easily defined. The main danger is that the definition can be so broad as to be useless. Intuitively, a user model is a collection of information a computer system keeps about a user. This information may be kept in a variety of forms, however. In fact, all computer programs have a user model, in that every program embodies an implicit model of the user, encoded by the programmer when writing the program. Wahlster and Kobsa [Wahlster 86] have proposed a definition that avoids such interpretations by requiring the knowledge of the user to be explicitly encoded. A slight re-wording of their definition is the following:[1]

> A *user model* is a knowledge source in a system that contains explicit assumptions on all aspects of the user that may be relevant to the behavior of the system.

The range of knowledge a system might keep about the user is purposely not specified in this definition. Depending on the system, a user model may be required to keep not only information about what the user knows or believes about things in the world, but also information about beliefs the user has about other people, the user's attitudes, capabilities, goals and plans.

"Assumption" as used in the definition above should be interpreted as a belief about the user, whether justified or not. Thus some assumptions in the user model may be well supported by evidence from the behavior of the user (or from other sources), while other assumptions may have no justification beyond the fact that someone placed that information into the user model (the system designer, for instance).

This definition will be adopted for now, although it is still quite broad. In practice, user models have been restricted in the range of information they can handle, or in the types of demands they are able to support. Nevertheless, a general definition is safer, otherwise, important aspects of user models might inadvertently be omitted.

[1]The original definition was presented in the context of natural language systems only. That definition has been expanded to include user models in any context.

2.2 Generality Criteria

Just as "user model" can be a slippery term, the use of "general" can lead to some confusion. There are actually three criteria of generality that are applicable to user modelling:

- *User Generality*: A user modelling system is general with respect to the users it can model if it can easily handle a variety of different types of users.
- *Domain Generality*: A user modelling system has domain generality if it can easily be transferred to operate in a system with a different underlying domain.
- *Interaction Generality*: A user modelling system has interaction generality if can function well under varying types of interactions with the user.

These criteria are discussed in the following sections.

2.2.1 User Generality

User generality has been addressed in existing user modelling systems. The main reason for developing user modelling systems is to enable systems to tailor their behavior based on knowledge of the specific individual currently using the system. To handle a range of possible users, three techniques have been used.

1. The range of possible information the user might believe is explicitly anticipated. This is the case with systems that incorporate a bug library [Johnson 84] or bad plan library [Sidner 81]. An individual user model is built by selecting some subset of this range of possible information. Such systems attempt to keep a list of all the possible information or plans the user might have.
2. The system may use a *generic* model. Frequently, system users will have a substantial amount of characteristics in common. For example, all users of a data base query system can be expected to have a goal of obtaining information from the system. This common information can be kept in the generic model. When a new user is encountered, the generic model provides a basis for an initial set of assumptions about the user that would otherwise be unavailable.
3. The system may employ *stereotype* models. Stereotype models are an extension to the idea of a generic model. Even when all users of the system do not share a large number of features, one may be able to categorize them into various classes of individuals. A stereotype model can then be used for each class of user. When a new user is encountered, the system classifies the user into one of these categories, to be able to use all the information contained in the stereotype for that class of user.

2.2.2 Domain and Interaction Generality

Previous user modelling systems have lacked domain and interaction generality. Most user models have been built to support specific applications. The model serves its purpose for the application, but its use in other applications is not considered. Often the user model is integrally linked to the rest of the system, such as the buggy procedures in DEBUGGY [Brown 78] or bug libraries in PROUST [Johnson 84]. A general user model should be able to work in a variety of contexts, a change of domain should not result in the loss of user modelling capabilities.

Most user models support only a specific form of interaction. In natural language systems, most of the focus has been on question answering [Allen 80,Kaplan 82,Carberry 83]. In intelligent tutoring systems, much of the study has concentrated on inferring plans by observing student behavior in arithmetic [Brown 78], algebra [Sleeman 82], or programming [Johnson 84,Murray 85,Reiser 85]. The effectiveness of the user model depends on the type of interaction with the user, and on the underlying domain.

Interaction with the user can take a variety of forms. It may be very restricted, such as a menu-based system on one with a simple command language, or it may be flexible, such as a natural language communication. The system might simply answer questions, or might be able to engage in mixed initiative dialogues with the user. In other situations the mode of communication could be visual or graphical.

The type of user interaction influences how the user model acquires its information about the user. If the interaction is in natural language, the system can use an internal representation of the statements made by both the user and system. In some forms of dialogue the system may even be able to ask questions of the user in order to augment the user model. On the other hand, an intelligent help system might be expected to acquire information about the user solely from observing how the user uses the system [Shrager 82,Zissos 85]. In the latter case, the user modelling system may have more information available to work with than in a natural language interface, but the difficulty in determining what the user is doing can also be much greater.

The type of interaction also influences how a system will make use of a user model. With an intelligent help system, the user model may be expected to detect misconceptions on the part of the user and "wake up" the help system so that it can decide whether to initiate a discussion with the user. Some interaction forms may require the user model to simulate the user. For example, a natural language interface may want the model to simulate the user in order to anticipate ambiguities the user might perceive, or misconceptions the user might hold as a result of the system's utterance.

2.2.3 Why General User Models?

A general user model is attractive because of the cost of building user models. All the user modelling systems cited above rely on a large amount of knowledge about the user pre-encoded by the system designers. Frequently the amount of knowledge required for the user model exceeds the amount of knowledge in the underlying application knowledge base. Acquiring this knowledge is not easy. Unlike the domain knowledge of the application, where the correctness can be judged, the information in the user model is largely a result of guesswork and intuition. For example, the intelligent tutoring system GREATERP [Reiser 85] tutors students learning the programming language Lisp. The underlying knowledge of the Lisp language was not hard to come by, but identifying the misconceptions and bad plans of students learning to program in Lisp required several man-years of painstaking effort.

2.3 User Modelling Requirements

A general user modelling system will have many requirements. These requirements can be classified into five categories: the content of the model, how it is used, how the model information is acquired, how the information is represented, and how the user modelling component interacts with the rest of the system.

2.3.1 Content of the Model

The content of the user model is the information about the user that may be useful to an application. Different applications have different requirements for information. If a user model is to be general, it needs to support any of these requirements. Thus a general user model must contain any type of information about the user that might be useful to an application. The potential contents of a user model can be classified into four categories: goals and plans, attitudes, objective properties and user beliefs.[2]

Goals and Plans A common problem for many applications is determining what the user really wants to achieve by using the system. This is particularly true of natural language systems, since the user frequently does not state his or her goal explicitly. Thus a user modelling system may be required to discern and keep information about the goals of the user. In many cases the user will have a plan for achieving these goals. Some applications, such as intelligent tutoring systems, need to know user plans in order to judge whether they are correct. This is also true for cooperative systems that seek to correct a user's plan if the system recognizes the plan will not achieve the user's goal [Pollack 85].

Attitudes People have definite opinions about certain things. For a system to interact with the user in a natural manner, it sometimes must be sensitive to the attitudes held by the user. For example, systems that advise the user must be cognizant of user attitudes when making a recommendation, both to avoid suggestions the user will not accept, and to reassure the user that particular requirements are being met [Pollack 82].

Objective Properties Many applications need to know objective properties of the user. Often these properties, such as the user's name or age, can be ascertained without talking to the user. Objective properties could also include the user's physical capabilities, such as whether the user is capable of performing a certain action or test recommended by the system.

Beliefs Most applications require some knowledge of the beliefs of the user. These beliefs encompass not only beliefs the user has about the world or about the domain of the application, but also the beliefs the user has about the system and about other agents. Tutoring systems and help systems have a great need for knowledge about the beliefs of the user, since they must be able to accurately judge what the user does and does not know. Cooperative systems in general also need such knowledge to help tailor responses made to the user, to ensure that what the system says is understood by the user and is not misleading.

2.3.2 Method of Use

A user model may be used either *descriptively* or *prescriptively*. The descriptive approach treats the user model simply as a data base of information about the user. An application queries the user model to discover the current view the system has of the state of the user's goals, plans, attitudes, objective properties and beliefs.

A user model is utilized prescriptively when it is used to "simulate" the user. For example, the HAM-ANS system [Hoeppner 83] employs the user model prescriptively during the generation of elliptical statements. HAM-ANS uses an *anticipation feedback loop* [Wahlster 86] to generate a

[2]A much deeper discussion of these categories appears in [Kass 86,Kass 87].

proposed response, then check that response against the user model to see whether the user would find it misleading or ambiguous.

How the user model will be used is an important point that is often overlooked. The working definition of user model calls it a knowledge source, corresponding to the descriptive use of the user model as a data base. A major motivation for revising the definition for "user model" in the next section will be to explicitly discuss how the model is to be used.

2.3.3 Method of Acquisition

The knowledge that a user model contains can be acquired in two ways: *explicitly* or *implicitly*. Knowledge is acquired explicitly when an individual provides specific facts to the user model. Explicit knowledge acquisition most often occurs when a system designer builds generic or stereotype user models. Knowledge can also be explicitly acquired from the user. Part of the user modelling system GRUNDY [Rich 79] uses this technique. When a person uses the system for the first time, GRUNDY asks for a list of words that described the user. This list is used as a basis for triggering predefined stereotypes to quickly build a robust model of the user.

Implicitly acquired knowledge about the user is obtained by observing the behavior of the user and the system. This may mean "eavesdropping" on the conversation between user and system, or may involve observing how the user uses the application. Implicit knowledge acquisition is difficult, but, as discussed in chapter 3, essential to produce effective user models that are domain independent.

2.3.4 Representation

Many issues arise in representing knowledge about the user. Most of these issues are shared with the general knowledge representation enterprise. Thus the problems of how to represent goals and plans, for example, are not unique to user modelling. However, two issues merit special consideration here. These are the issues of representing beliefs about multiple agents, and the non-monotonic nature of belief modelling systems.

User modelling is really subset of a larger field called *agent modelling* [Kobsa 86]. Agent modelling is concerned with building models of entities a system might need to know about. These entities need not be human, they might be other computer systems, for example. A situation where multiple agents need to be modelled is a doctor consulting an expert system about the treatment of a patient. In this case the doctor is the user, but the expert system is primarily concerned with modelling the patient to determine what treatment to suggest.[3] A user model is thus an agent model for the individual currently interacting with the system.

A general user model must be able to represent the beliefs a user has about the beliefs of other agents. These beliefs can be arbitrarily complex. For example, it may be necessary to represent a belief the user has about a belief the system has about the user, or even more deeply nested belief structures. A special case of this situation is when the nesting is infinite. This occurs when representing cases of mutual belief [Joshi 82], where two parties, call them S and U, believe a fact p (which we write SB(p) and UB(p) for S believes p and U believes p respectively), and further believe the other believes it (SBUB(p) and UBSB(p)), and believe the other believes they believe it (SBUBSB(p) and UBSBUB(p)), and so on. (Figure 2.1 illustrates the modelling S and U have

[3]This example is taken from [SparckJones 84] and presents an unfortunate conflict in terminology. Sparck-Jones distinguishes between *patient* and *agent* models when discussing interaction with an expert system. The patient is the object of the discussion, in this example the medical patient. Sparck-Jones calls the individual directly interacting with the system the agent. Thus both Sparck-Jones's patient and agent models are agent models in Kobsa's terminology, while Sparck-Jones's agent model corresponds to Kobsa's user model.

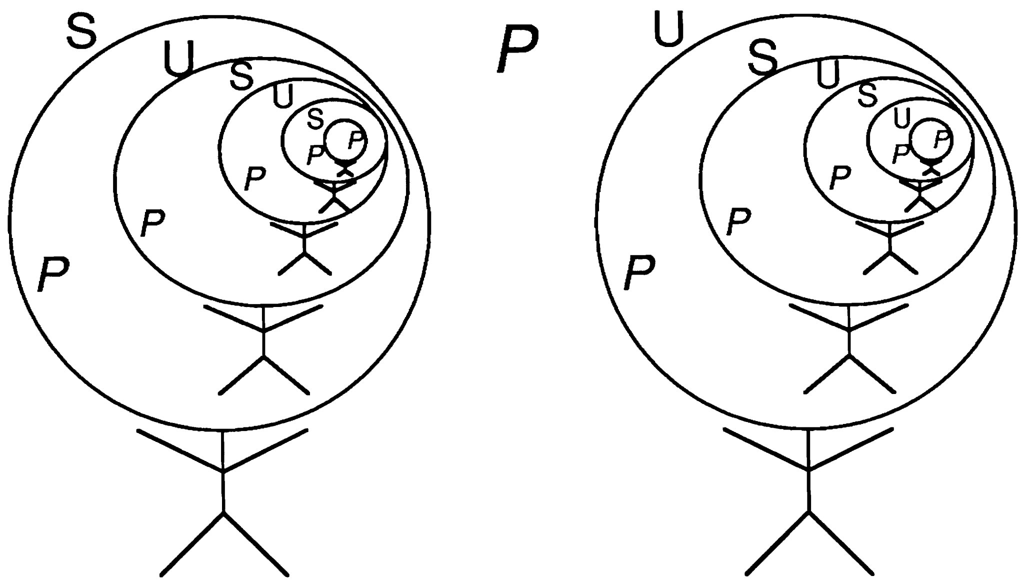

Figure 2.1: S and U's modelling of the mutual belief of p

of the mutual belief of p.) Representing such arbitrary nestings of beliefs about the the same underlying basis of facts (the p's) is a problem unique to agent modelling.

The second representation issue important to user modelling is the inherent non-monotonicity of the information in the user model. Beliefs about the user that the system holds can change in two ways. First, the user modelling system may make an assumption about the user. Later, the system may discover the assumption is incorrect, and hence have to change the model, retracting the old assumption and adding a new one. Secondly, the user model may change simply because the user changes. During the interaction a user may learn information he did not know previously, or that corrects a misconception he had. No matter what the cause, the user modelling system must be able to deal with a model in which much of the information can change with time. The system must have effective ways of juggling alternative models of the user and changing assumptions when necessary.

2.3.5 Communication with Other Modules

Not only does the form of interaction with the user influence the user model, the user model's interaction with the rest of the system is important. Besides issues such as the communication language used and protocols for interaction, there is one important distinction in the types of inter-module communication.

A user model may be either *passive* or *active*. A passive user model only *responds* to information and requests. On the other hand, an active user model may, on its own initiative, volunteer information to another module.

An active user model can be important because of the non-monotonicity of the knowledge in the user model. Suppose a module M requests information from the user model. The user modelling component U responds that currently the item p is believed. Later, new information is acquired that causes U to retract belief in p. If U is an active user model, it can volunteer to M the information that p is no longer believed, and perhaps provide new information in place of p.

2.4 Revised Definition of User Model

The original definition for "user model" is flawed. The notion of the user model as a source of knowledge about the user is adequate for explaining what information a user model should contain, but cannot account for the ways the model is used, or the ways it interacts with the other modules in a larger system.

2.4.1 Problems with the Initial Definition

The original definition is inadequate because it fails to include the notion of "model." A model can be thought of as a partial image of the thing being modelled. A model enables analysis of an entity, or prediction of its behavior without directly manipulating the entity. In this context a user model is able to act as a substitute for the user. It should be not only a source of information about the user, but also be a "little copy" of the user that the application can use when reasoning about the user.

The problem with the original definition is due to an incorrect distinction between the user model itself, and the machinery that manipulates the user model. Commonly, the user modelling portion of a system is divided into the *user model* itself, and a *user modelling component*. The user model is the information about the user, and nothing else. The user modelling component can be viewed as a kind of super data base or knowledge base manager. The user modelling component is the software that maintains the user model, acquires information for the model and supports information requests from other modules in the system.

The distinction between a user model and the user modelling component is valid, but the line between the two has been drawn in the wrong place. The easiest way to illustrate this point is with an example.[4] Suppose there are two individuals, S and U, and S wishes to communicate a piece of information to U.[5] To do so, S formulates an idea of what to say, call this p (see figure 2.2). Before uttering p, S will first consider what effect p will have on U, to make sure that U will understand p, and that U will not draw any incorrect conclusions because of S saying p. This step requires S to look at its model of U, and "try out" saying p to the model of U (figure 2.3). When U hears S say p, U interprets p with respect to the model U has of S (figure 2.4). Consequently, for S to "try out" p on the model of U requires S's model of U to reason as U would when U hears p.

The point of this example is that the model S has of U must have strong reasoning capabilities, including knowledge acquisition capabilities and the ability to reason about multiple models. Knowledge acquisition until now has been considered part of the user modelling component, the support software of the user model. The reasoning used to interpret an utterance normally exists in the application. As the example illustrates, these forms of reasoning belong in the user model itself.

The user model versus user modelling component distinction has a parallel in expert systems. Here the difference is between the domain rules of the expert system, and the control of the reasoning process of the expert system. In expert systems, knowledge of the domain itself is distinguished from *meta-knowledge*, knowledge about knowledge [Davis 84]. In user modelling the distinction is between the goals, plans, attitudes, objective properties and beliefs held by the user, and the knowledge used to reason about this information.

[4]This example is an instance of a speaker attempting to prevent false inferences a hearer might make. See [Joshi 84] for details of this issue.

[5]Assume a natural language form of interaction for this example.

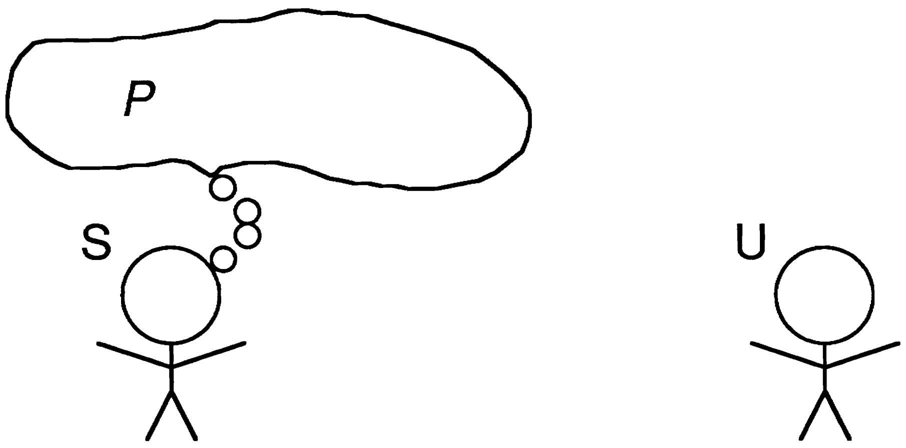

Figure 2.2: S considers saying p

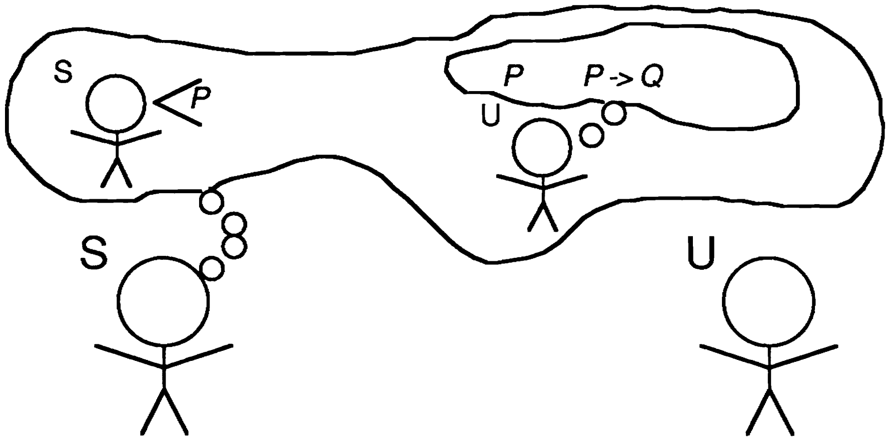

Figure 2.3: S "tries out" saying p to U

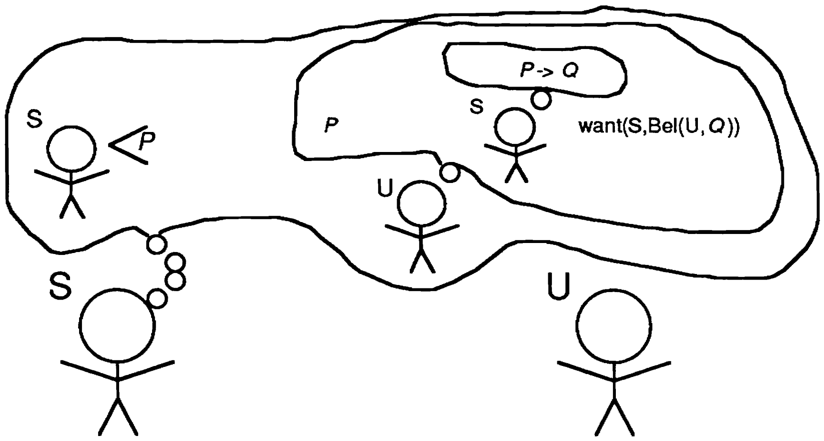

Figure 2.4: S examines the expected effects of p on U

2.4.2 The Revised Definition

With these criticisms of the initial definition in mind, a new definition for "user model" can be proposed.

> *A* user model *is an explicit collection of beliefs the system has about the goals, plans, attitudes, objective properties and beliefs of the user, together with beliefs the system has about the meta-knowledge and reasoning employed by the user in manipulating that information.*

The inclusion of meta-knowledge in the user model addresses the problems raised in the previous section. The meta-knowledge in the user model gives it the ability to reason about its own knowledge (including the reasoning knowledge itself). Thus the user model can be used to simulate the user, making it truly a "model." At the same time the user model is more than simply a knowledge source, since the model now has an active reasoning capability. Finally, the meta-knowledge that was hidden in the user modelling component or in the application has now been placed in the user model itself, where it belongs.

2.5 The General User Modelling Dream

Every researcher should keep in mind some long range goal toward which his or her current work is leading. It is useful from time to time to stop and look once again at that long range goal, if only to put the work into perspective and to help focus efforts. In this case, since the field is so new, and so much work needs to be done, the goal is more of a dream.

The general user modelling dream is simply this: to have a user modelling module that can be used effectively and easily by any system that could use a user model. This means that the modeller can support systems spanning the three dimensions of generality mentioned earlier: it will be able to deal with any kind of individual, support any form of interaction with the user that the system may use, and be effective, regardless of the underlying domain of the application.

A completely general user modelling component can thus be thought of as an independent module functioning on its own behalf. The other modules of the system have no concern with how the user modelling module functions, they will simply use it as needed. A module-based approach means a well defined interaction language must be available for communicating with the user modelling module. This language defines the set of functions the user modelling component will support, as well as the requests a user modelling component can be expected to make. These interfaces will be at a general level, abstracted from domain or interaction specifics. A general user modelling module is thus a utility, one of several building blocks used in the construction of an overall system.

A completely general user model really is a dream. The range of situations in which such a system might operate extends from children using a mouse to draw pictures, to pilots controlling an airplane through combinations of voice and body movements, to an order entry system, to a system that gives advice about cooking dinner. Such a range of capabilities is beyond that of humans. Still, stretching the imagination helps when it aids analysis of the problems to be faced.

2.6 Terminology

This is a good point to clarify some of the terminology that has been and will be used in this paper. A variety of terms exist for referring to various aspects of user models or user modelling.

Frequently writers will use these terms interchangeably, resulting in some confusion for the reader. To avoid this, a short dictionary of user modelling definitions is presented here.

Individual User Model The information a system keeps about a specific user of the system.

User Model The formal definition is given earlier in this chapter. In practice, a user modelling system often will not keep information specific to each user of the system. Rather, generic or stereotype models are kept. Hence "user model" often refers to all the information the user modelling system has about the system users. This is the sense in which it will be used here. "*Individual* user model" will be used to refer to the model that applies to a specific user.

User Modelling Component The software that maintains the user model and handles the interface of the user model to other components in a system.

User Modelling Module The user model together with the user modelling component. The user modelling module is simply that portion of the system that "does" the user modelling.

User Modeller One who does user modelling. Generally this refers to the individual or individuals who explicitly encode information for a user model.

Chapter 3

Acquiring User Models

The acquisition of user models is an important and interesting problem. Many authors have pointed out that the availability of a rich user model would enhance the performance of their systems, particularly in the area of natural language systems.[1] The major problem in making such user models available has been the difficulty in obtaining the knowledge that belongs in the user model. User model acquisition is interesting and challenging in a general sense because it touches on a number of fields important to Artificial Intelligence. These include knowledge representation, non-monotonic reasoning, cognitive modelling, and perhaps learning.

A major distinction exists between techniques that emphasize acquiring information explicitly, and methods that emphasize implicit acquisition. Much of the emphasis in user model acquisition to date has been on the explicit acquisition of model information. This approach will be reviewed, and its major difficulties noted. On the other hand, implicit model acquisition has not been well studied. The second section will focus on the characteristics of implicit model acquisition. The goal of this work is to demonstrate that implicit user modelling can be a powerful, general method for acquiring information about the user. Hence, the remainder of this paper will concentrate on the task of acquiring user models implicitly. Figure 3.1 presents a taxonomy of the user model acquisition techniques that will be discussed.

3.1 Explicit Model Acquisition

User models may be built either during or prior to interaction with the user.[2] GRUNDY is an example of a system that explicitly acquires information during an initial interaction with a new user. In many cases this method of acquiring information is not a viable alternative. Brown and Burton [Burton 82b] have observed, in modelling students who have little knowledge of the domain, that direct questions about the student's knowledge are not helpful because the student doesn't

[1]McCoy [McCoy 85] has expressed the need for acquiring extensive knowledge of the user to support the correction of user's object-related misconceptions. McKeown, in describing future work on tailoring explanations for the user, says [McKeown 85, p. 798]

> It seems likely, also, that better explanations will require a more complete user model incorporating static, global characteristics of the user as well as those dynamic, local characteristics available from the ongoing dialogue itself.

[2]Actually, a user model could be acquired *after* interaction with the user occurred. For example, the model could be built from a history of the user's interaction with the system. Apart from issues of computational efficiency and time limitations, this form of acquisition is equivalent to acquiring the information during the interaction with the user.

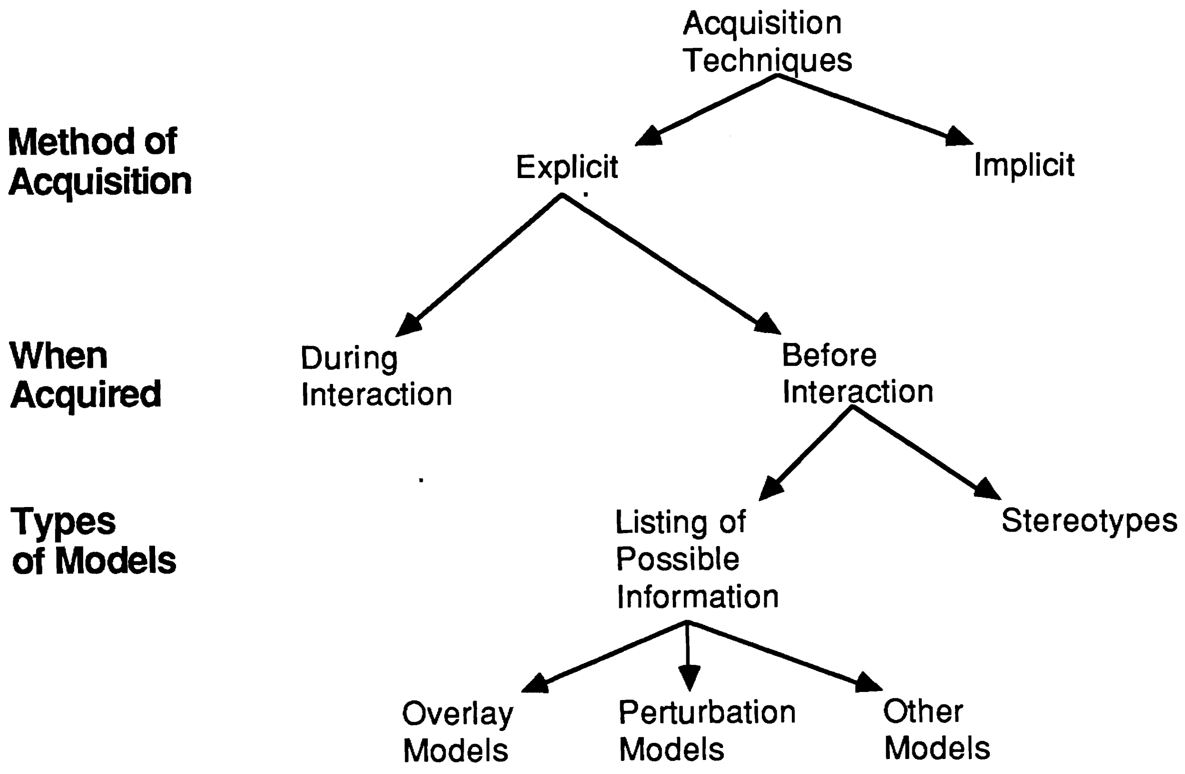

Figure 3.1: Taxonomy of User Model Acquisition Techniques

know enough to correctly answer the questions. If the system instead asks indirect questions, in an effort to infer the student's knowledge, the number of questions may quickly become excessive. Direct query of the user is also not viable in most natural language interactions, because such questions are often considered socially unacceptable.

Because explicit model acquisition during an interaction is frequently impossible, the primary emphasis has been on acquiring information before interaction with the user. Acquiring user modelling knowledge before interaction with the user simply means that the knowledge in the user model must be hand-coded by an individual before the model is used. Section 2.2.1 discussed three techniques that may be used:

1. Encode the *range of possible information* a user might have.
2. Encode a *generic* model to represent the information expected to be held in common by all users.
3. Encode *stereotypes* that contain information expected to be held in common by classes of users.

Generic user models are quite similar to stereotype models. In fact, a generic user model can be viewed as an instance of a stereotype model, in which only one class of users is considered. Thus generic models will be included in the discussion of stereotype models in the remainder of this section.

3.1.1 Models that Specify a Range of Information

User models that encode a range of information may have a variety of forms. The basic approach is to pre-encode a collection of information that may be applicable to system users. During interaction, the user modelling component observes user behavior to try to identify pieces of pre-encoded information that apply to this particular user. Implicit model acquisition occurs, in that the user model for the specific individual is built during the course of interaction, but this implicit modelling is limited to selecting previously encoded information that best fits the current user. Thus the explicit modelling activity is one of *recognition*. In fact, a recognition technique is used by all methods that emphasize implicit user model acquisition. The recognition process may be one of two types: the pre-encoded information may be a list of competing possibilities, of which only one may apply to the user, or some subset of the pre-encoded information may apply to the specific user.

Selecting from Competing Possibilities

Plan recognition is an example of selecting from a list of competing possibilities. Many systems assume that the user is motivated by a single plan to accomplish some goal. The task of a user modelling system is to identify what plan the user holds. This technique is used frequently in intelligent tutoring systems. For example, PROUST [Johnson 84,Johnson 85] has a library of possible plans (both good and bad) a novice Pascal programmer might possess. When the student compiles a program, PROUST tries to identify which plan the student is following. Similar approaches are used in GREATERP, a Lisp tutor [Reiser 85], the Geometry tutor [Anderson 85] and TALUS, a Lisp program debugger [Murray 85]. In question answering systems, this technique for plan recognition has been used by Allen and Perrault [Allen 80] to recognize plans of users at a simulated train station information booth.

Selecting from a list of competing possibilities has limited applicability. Even in the context of plan recognition, recent systems, such as ARGOT [Allen 82] and TRACK [Carberry 83], have emphasized the need to recognize multiple plans that a user holds concurrently. In most situations, many items of information about the user are important, so choosing only one is not possible.

Selecting a Subset of Information

The user modelling component may also need to recognize a subset of the pre-encoded information. In this situation an individual user model is built by using the behavior of the user to select which information applies to him. This process can be much more complex than simply selecting among alternatives. Since the process is incremental, each new action of the user can potentially cause the selection of a new item for the individual user model, with no way of determining a termination point.

Overlay user models are a classic example of a technique that views the user model as a subset of some other collection of information. In the case of overlay models, the pre-encoded information is the domain knowledge of the underlying application employing the user model. Figure 3.2 illustrates an overlay model. Overlay modelling was used by Carr and Goldstein in WUSOR-II [Carr 77], an intelligent tutoring system for advising users playing the game "Hunt the Wumpus." The domain model for WUSOR-II was the knowledge base for an expert Wumpus player. An overlay model is also used by GUIDON [Clancey 82], an intelligent tutoring system for tutoring medical students in diagnosing cases of meningitis.

Overlay modelling suffers from a serious drawback. An overlay model assumes that the knowledge a user has is a subset of the knowledge the system has. Thus an overlay model cannot correctly

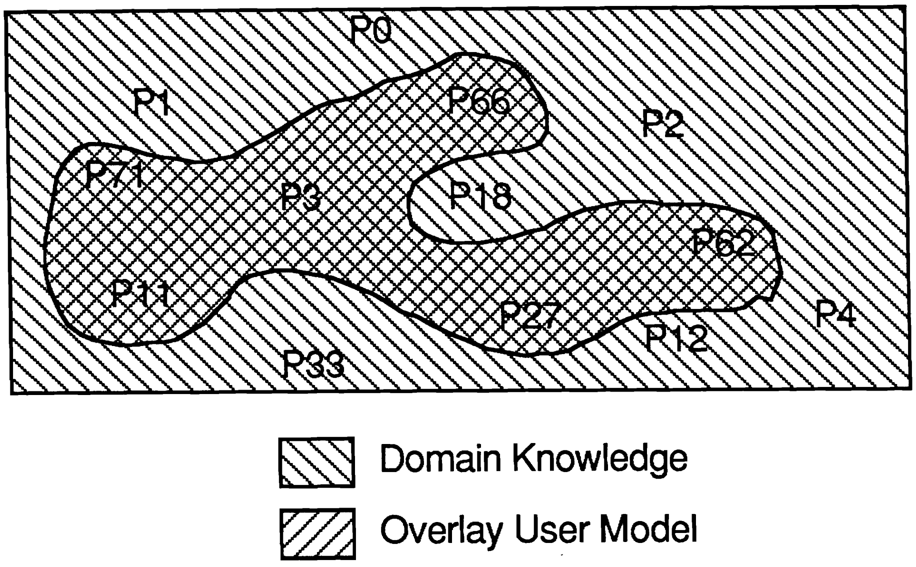

Figure 3.2: An overlay user model

model a user whose behavior is motivated by information the system does not have. Even more problematic, the overlay model cannot recognize misconceptions a user has about information in the domain model. For example, if the user believes that whales are a kind of fish, an overlay model will not be able to record that fact.

Because of the drawbacks of the overlay model, an extension called a *perturbation* model has frequently been employed. The perturbation model still uses the domain knowledge of the application as a basis, but augments this knowledge with additional information, such as misconceptions, which the user might believe. The intuition behind this sort of model is that the user's beliefs may differ from the domain model, but not radically so. The individual user model is thus close to the domain model, with some minor perturbations. Figure 3.3 illustrates a perturbation model.

Perturbation models have been used extensively. A good example is the student modelling in DEBUGGY [Brown 78,Burton 82a]. DEBUGGY seeks to explain the procedural errors of students performing multi-column subtraction. Extensive analysis of student test results enabled them to build a *skill lattice* containing correct and incorrect skills that might be used in performing a multi-column subtraction. The performance of any given student was then modelled as a path through this skill lattice.

Problems with Range of Possibilities Modelling

All of these approaches share some common problems, including: brittleness, lack of domain generality and slowness to adapt to a new user.

Brittleness The success of range of possibility modelling depends crucially on the information that is pre-encoded. If the user model builders omit information that a user might believe, and that is important to the application, the performance of the user modelling module and of the application will be impaired. Supplying this information is a very difficult task, since the model builder must anticipate all possible system users.

The brittleness problem has been recognized for some time. In dealing with this problem, Brown and VanLehn [Brown 80] have developed *repair theory*, in an attempt to provide a generative theory

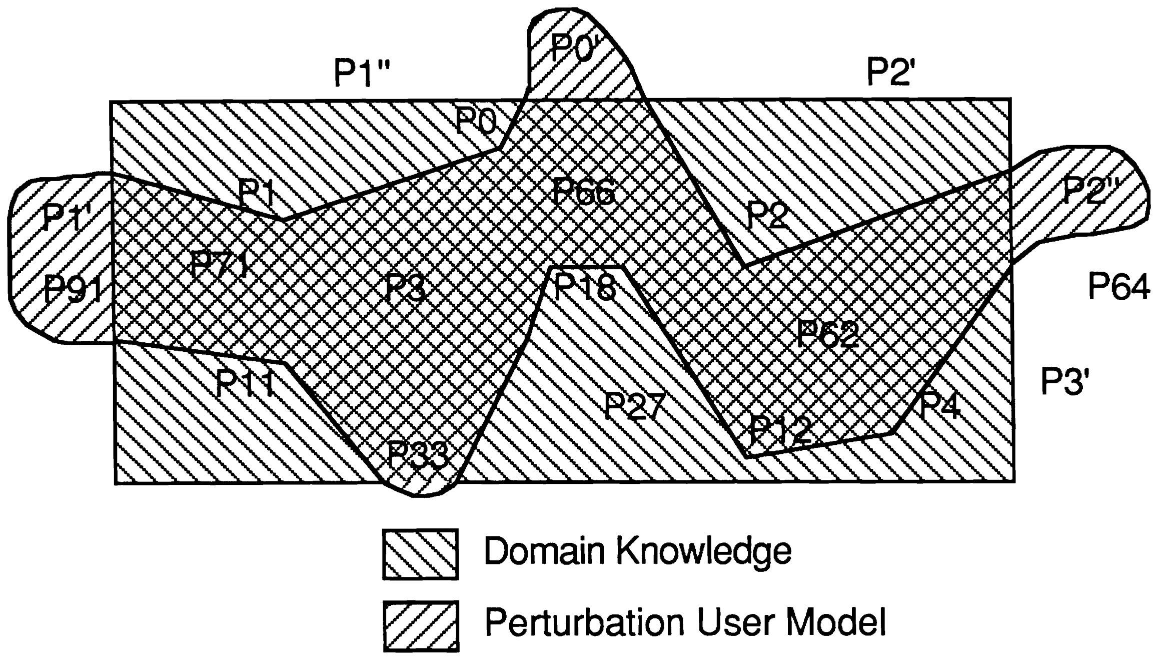

Figure 3.3: A perturbation user model

of the bugs students have about arithmetic. By "generative" they mean the theory is capable of simulating the behavior of the student on problems the student (and system) has not seen before. Repair theory still keeps the idea of a perturbation model, but rather than pre-encode all the "buggy" rules students might have, Brown and VanLehn propose a set of *repair heuristics* to explain the student behavior. They suggest that a student proceeds to solve a problem using rules previously learned. When a student encounters a situation s/he doesn't know how to handle, the student attempts to "repair" the problem to get back to a situation that can be handled. Brown and VanLehn propose that these repair heuristics are small in number and commonly shared by all individuals. Repair heuristics can be used to avoid the brittleness problem, but are quite restricted in domain themselves.

Lack of Domain Generality With the exception of the overlay model, changing the modelling domain require a substantial re-coding of information. An overlay model would be flexible in this regard, since it could simply use the new domain knowledge as the basis for overlaying user models. However, overlay models have their own problems, mentioned earlier. The amount of re-coding required for a change in domain depends on the set of information that is in the user model, but not in the domain model. Unfortunately, this set is often quite large in practice.

Slowness to Adapt For a system to quickly adapt its behavior to a new user, a robust user model is needed. The third problem with range of information modelling is that building individual user models can be very slow. Much of this slowness is due to the simple recognition techniques used. Typically, the implicit acquisition techniques are limited to simple rules such as:

If the user mentions C, then the user knows about C.

Rules of this sort don't make enough inferences about the user to build a robust model quickly. For example, suppose the user mentioned the term "windows" in talking to an intelligent help system for the EMACS editor. The simple recognition methods would add the fact that the user knows about the term "window," but would fail to add a whole host of other information a human expert might assume the user believed because he used that term. In particular, a human expert might assume the user was more advanced than a novice, since knowing about windows isn't essential to using EMACS, that the user understands the concept of editing multiple files concurrently, and perhaps that he knows about buffers. The slowness to adapt of the user model is one of the major motivations for implementing stereotype models.

3.1.2 Stereotype Models

Encoding stereotypes in a user model provides an extra level of structure over simply listing a collection of possible information. With stereotype models, the list of information is still present, but is organized into units that are expected to more accurately model classes of users. The notion of a stereotype is intuitively appealing. Humans seem to use stereotyping to quickly categorize new people they meet. Since humans seem to do a good job of modelling other individuals, stereotyping appears to be a promising technique to employ in user modelling as well.

Stereotype modelling enables a system to develop a large set of beliefs about a new user very quickly, by selecting a stereotype (or stereotypes) that characterize the user early in the interaction. Two techniques have been used. The first is the use of *triggers*, introduced by Rich in GRUNDY [Rich 79]. Associated with each of the stereotypes in GRUNDY are one or more triggers. If the user mentions a trigger item, the entire stereotype is invoked. Thus a few pieces of information about the user can potentially cause the system to hold a large number of beliefs about the user.

The second technique for selecting stereotypes uses a matching or "best fit" approach. Using this method, a stereotype that best explains the user's behavior so far is selected. Such an approach also allows stereotypes to be refined or even changed as new information is acquired. An example of this approach is GUMS$_1$ [Finin 86], which organizes stereotypes into a hierarchy. Stereotypes lower in the hierarchy inherit information from those above, in addition to having more specific information as well. Selecting a stereotype for the user can thus be viewed as a *classification* process.[3]

3.1.3 Problems with Stereotypes

The stereotype approach to user modelling suffers from several difficulties. Like the method of listing potential information about the user, stereotypes are prone to brittleness and lack domain generality. They have other problems as well. The first problem only occurs with a particular form of stereotype model called a *level stereotype*. Although these models are intuitively attractive, they are frequently not appropriate. A second problem is that stereotype models can be very difficult to build. Finally, reasoning based on stereotype models lacks adequate justifications.

Inappropriateness of Level Stereotypes

A common stereotyping method is to classify users according to a range of *skill levels* with respect to the underlying domain. For example, users of an operating system might be classified as novice,

[3]It is a classification process in two senses. Since stereotypes hold for various classes of users, the information obtained from the behavior of the user is used to determine which class the user belongs to, and hence which stereotype is applicable. It is also a process similar to the knowledge base classification used in KL-ONE [Brachman 85b,Lipkis 82,Schmolze 83]. Here classification is based on the notion of subsumption. A subsumes B if and only if $B(x) \Rightarrow A(x)$ for all x. A correct classification is accomplished by "pushing" an object down in the hierarchy as far as will go, based on testing subsumption relations at each level.

beginner, intermediate or expert, as is done by KNOME, the user modelling component for the UNIX Consultant [Chin 86]. Using level stereotypes presumes that users learn information about the domain in a uniform manner. If the domain is such that users can become specialists in particular subdomains, level stereotypes will be inappropriate. For example, a new UNIX user might be curious about pipes, and thus learn a lot about them. His knowledge of pipes would imply the user was an intermediate or expert at using UNIX, when in fact he is still a beginner. Level stereotypes are used frequently because they are fairly easy to implement, but often user knowledge does not correspond to the "level hypothesis."

Difficulty of Building

Stereotypes are difficult to build—more difficult than the "collection of possible information" approach. Stereotypes require not only that information about all potential users be encoded, but also that this information is organized into the proper stereotypes. Thus building stereotype models is more difficult than building a domain model for an application. Not only is more information required for the user model, each stereotype must be correctly organized, a task comparable to building several related domain models.

Lack of Justifications

Stereotype models cannot justify why a particular item belongs in an individual user model, other than to note that a stereotype containing the item has been invoked. Thus the justification for an item's presence in the individual user model may depend on a completely unrelated fact that triggered a stereotype, or it may be that this item is contained in a stereotype that happened to best explain the information available about the user.[4]

Justifying the presence of information in the user model is important because of the non-monotonicity of this information. Often the arrival of new information about the user creates conflicts in the beliefs in the user model. Suppose two stereotypes are invoked that together hold inconsistent beliefs about the user. Rich dealt with the problem by using evidential reasoning in GRUNDY [Rich 79]. An evidential reasoning approach is unsatisfying however, because of the number of possible places where evidence might be applied. For example, evidential weights might be affixed to each of the following items:[5]

- The degree of belief in the trigger item
- The degree of belief in whether the trigger item should trigger the stereotype
- The degree of belief in the stereotype itself
- The degree of belief in whether accepting the stereotype implies that an item in the stereotype should be believed (this could be different for each item in the stereotype)
- The degree of belief in an item

[4]Thus the presence of an item may depend on the fact that *no other* stereotype was a better fit, a very poor justification.

[5]This problem is really a special case of a more general problem that arises when several reasoning steps occur between an explicit observation and a conclusion. Suppose a reasoning system draws two conclusions that are inconsistent with each other. If the chain of reasoning for each conclusion is not kept as a justification, the system has no way of determining which reasoning step (or steps) might be in error. Evidential reasoning systems suffer from this problem, because once an evidential weight is computed for a conclusion, there is no way to recover the reasoning that led to the conclusion.

It is hard enough to build stereotypes in the first place, applying even some of these weights would be excessive.

3.1.4 Stereotypes are Like Procedural Knowledge Representations

A parallel exists between stereotype user modelling and procedural knowledge representations. This parallel helps to explicate both the advantages and drawbacks of using stereotypes in modelling users. Three points in particular illustrate the similarities between stereotype user models and the procedural representation of knowledge.

Efficient Procedural implementations are usually much faster than comparable declarative implementations. Likewise, a user modelling system that employs stereotypes can more quickly develop a large number of beliefs about the user than a system that does not employ this additional structure.

Compiled Procedures usually represent a group of information that has been collected into a single execution unit. Likewise a stereotype is a collection of beliefs about the user that have been collected into a unit.

Lack Clear Semantics The criticism of procedural representations is that the only way to tell what they do is to run them. On the other hand, a declarative representation is not only executable, its meaning can be ascertained by looking at it. In the context of stereotypes, the lack of justifications for particular items in the stereotype plays a similar role. It is not possible to look at a single item in a user model and determine whether it should apply to a particular user.

3.2 Implicit Model Acquisition

Explicit user model acquisition techniques suffer from many problems. From the standpoint of general user modelling, the greatest is the need to encode new user model knowledge for each situation in which the model will be used. This severely limits the flexibility of the user modelling module, particularly since the explicit acquisition of model information is so difficult. If progress towards the dream of truly general user models is to be made, implicit model acquisition techniques will be needed.

Implicit user model acquisition has not been well studied. There are several reasons for this. First, the field of user modelling itself is relatively new (most of the work has been done in the last ten years), and hence many areas have not been explored yet. More importantly, implicit user model acquisition has not seemed promising. The success of implicit user modelling is strongly constrained by two factors: the seeming lack of information that can be used to build a model of the user, and the lack of certainty in any implicit acquisition techniques. These limitations imply that any user model that relies on implicit acquisition techniques would be too slow in adapting to new users, and frequently wrong at that.

The goal of this work is to show that implicit user model acquisition *is* a promising technique, one that can enable effective user models as well. I hope to show that the limitations on information obtained about the user is not too great, at least in many situations, and that a lack of certainty in the acquisition techniques can effectively be dealt with. This section characterizes implicit user model acquisition by discussing the sources of information available for a user modelling module, and the incremental nature of implicit user model acquisition.

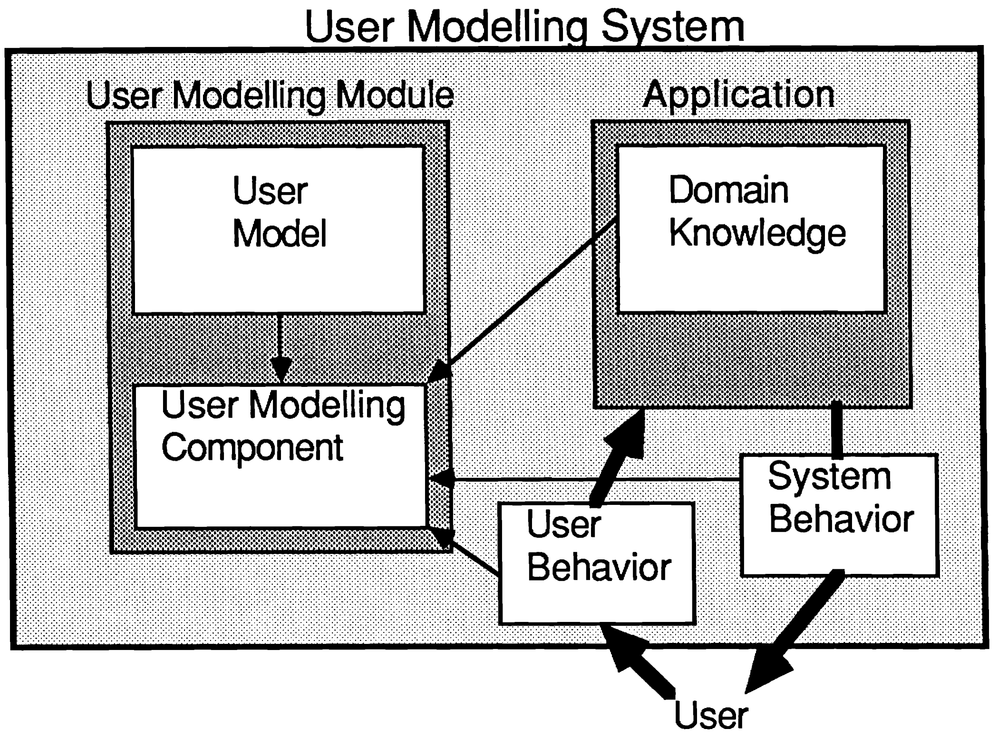

Figure 3.4: Sources of information for the user modelling module

3.2.1 Source of Information

In general, a user modelling module has two sources for acquiring information about the user: observable behavior of the participants during an interaction, and the existing knowledge of the system. Figure 3.4 diagrams the relation of the sources of information to the user modelling module.

Observable Behavior

The primary source of information for a user model is the observable behavior of the user. Generally this behavior is the actions taken by the user when interacting with the system. For a natural language system, the source of information about user behavior is the actual statements made by the user, perhaps as provided by the parser. In other situations the user modelling module may have access to the user's use of the application to perform tasks, such as the user's interaction with an operating system or editor.

Another source of information is the system behavior observable by the user. Actions made by the system that the user observes will affect the user's beliefs. At the very least, system actions should cause the user's model of the system to be modified. System actions can also be expected to cause a user to revise beliefs previously held. This is particularly true in situations where the user approaches the system for information or advice. The response given by the system is information sought by the user, and hence information that should modify the user's beliefs.

Existing Knowledge

A large source of information for the user modelling module is the domain knowledge of the underlying application in the system. The overlay modelling technique is an example of this. The domain model serves as a basis for interpreting the beliefs and intentions of the user. A more sophisticated technique is the *differential model* approach. Differential modelling uses the domain model to simulate what the system would do in the same situation as the user, in order to identify not only the things in the domain the user knows (as in overlay modelling), but also things the user *should* know (according to the system's simulation) but doesn't. Differential modelling is used in the WEST tutoring system [Brown 75,Burton 82b] where students play a board game in which players' moves depend on certain arithmetic skills. The expert (system domain model) always knows the best move and thus can detect lack of knowledge on the part of a student who consistently makes inferior moves that can be attributed to the lack of a particular arithmetic skill.

Another source of information is the user model itself. Recognition-based modelling uses the collection of pre-defined information in the user model as a source for selecting which information applies to the particular user. The current individual user model can also be important in interpreting new information about the user, and hence a source of information for the future user model. This is definitely true of the modelling people do. A person's current beliefs about an individual strongly influence any future beliefs they might hold.

3.2.2 Incremental Acquisition of the Model

Implicit user model acquisition is almost always incremental. Although some systems do implicit model acquisition in a non-incremental fashion,[6] this is usually not the case. In general, the user model is present in a system to enhance the interaction capabilities. This means that the user modelling module must be actively acquiring the user model as interaction progresses, to be able to support requests for information the system can use to improve its interaction.

The incremental nature of the acquisition process means the user modelling module must cope with incomplete information. At any given moment another module in the system may request a piece of information to which the user modelling module must respond to the best of its ability. The fact that the user model is continuously changing means the user modelling component must have some way of managing the possible choices for beliefs about the user. A truth maintenance system (TMS) [Doyle 79] is a good way to support these requirements. The TMS maintains the current beliefs held by the system (in this case beliefs about the user), along with justifications for those beliefs. As new information is learned the justifications serve as a basis for revising beliefs held by the system.

The fact that acquisition is incremental also simplifies the implicit model acquisition task. As new information arrives, the user model does not need to be re-built from scratch. Rather, the existing model is modified to incorporate this new information. In most cases the addition of this new information will cause only minor changes to the overall model. Thus incremental acquisition restricts the amount of work a user modelling module must do.

[6]An example is a user modelling module that uses transcripts of interaction between the user and the system to build a model of the user. The Macsyma Advisor [Genesereth 79,Genesereth 82] is such a system. The Macsyma Advisor analyzes a transcript of the user's session with Macsyma to generate a user model when the user asks for help.

Chapter 4

Cooperative Advisory Systems

The purpose of this work is to pursue implicit user model acquisition techniques that support a general user modelling module. Unfortunately, such a task is extremely difficult. The requirements of interaction and domain generality[1] combine to make completely general user model acquisition at present seem an insurmountable task. In light of this difficulty, the rest of this paper will focus on implicit user model acquisition where the form of interaction has been restricted to *cooperative advisory systems* that use a natural language interaction. The remainder of this chapter discusses why the form of interaction was restricted (as opposed to restricting the domain), and why cooperative advisory systems were chosen. The final section discusses what type of information will be considered in modelling the user.

4.1 Why Restrict Interaction?

The form of interaction will be restricted because implicit user model acquisition is very dependent on the type of interaction used. Varying the form of interaction can drastically alter the forms of information representing the behavior of user and system. Since the implicit user model acquisition process is driven by the information obtained from the behavior of the user and system, different forms of interaction necessitate different model acquisition methods. These techniques for model acquisition do not share much in common. For example, there is little overlap between the techniques used to acquire knowledge about the user in a natural language system, and in those used in a system where the user modelling module learns about the user by observing how the user interacts with the underlying application. At least for now then, acquisition generality across different forms of interaction does not seem likely. Thus limiting the form of interaction will restrict to a reasonable size the range of acquisition techniques that must be considered.

A second reason for focusing on domain generality rather than interaction generality is the existing trend in this direction. A good example of this is the field of expert systems. The expert system shell serves to restrict the forms of reasoning the system can perform, as well as the forms of interaction with the user. This shell can then accept, and perform well with, varying domains of information in its knowledge base. Focusing on domain independence in user model acquisition will thus enhance the capabilities of other systems that attempt to achieve domain generality. A domain-general user modelling module can be incorporated into other domain-general systems to enhance the interaction with the user.

[1]I am taking as given that user generality is a requirement for any user modelling system.

4.2 Why Cooperative Advisory Systems?

There are several reasons for restricting the range of interaction to cooperative advisory systems. Following a brief description of these systems, the reasons for focusing on cooperative advisory systems will be discussed.

4.2.1 Cooperative Advisory Systems

Cooperative advisory systems are systems that seek to provide advice to the user about a domain in which the advisory system has a body of expert knowledge. The systems are cooperative in that they try to be as helpful as possible: by providing explanations, correcting user misconceptions, trying to satisfy goals of the user that have not been directly stated, and so on. Clearly a user modelling module can be useful to such a system.

An advisory system has a mixed initiative interaction. Usually a user initiates the interaction by expressing a problem situation and asking the system for help. The system may then begin to ask a series of questions of the user while reasoning about the problem, finally giving its solution or advice. At any point the user may interrupt to question the system's reasoning or provide additional information. Once a solution has been given the user may wish to explore related alternatives, or to qualify the original goal.[2] Thus an advisory system differs significantly from a question answering system, since question answering systems generally do not take initiative in a dialogue.

An advisory system will usually have an expert system as the underlying application. The expert system is usually goal-directed: it will have a specific goal (recommending some form of advice about the domain) and then proceed to reason about the problem, requesting information as necessary. Both knowledge of the structure and components of the domain, and the reasoning used by the expert system are available in the application knowledge base.

4.2.2 Advantages of Studying Cooperative Advisory Systems

There are several reasons for focusing on user model acquisition in cooperative advisory systems. These reasons can be divided into two classes: reasons why this form of interaction enables better model acquisition techniques, and general reasons why user modelling in such systems is interesting, useful and still quite challenging. The following paragraphs present some of the primary reasons for studying user model acquisition in the context of cooperative advisory systems.

Accessible Information

A major advantage of cooperative advisory systems that communicate in natural language is the accessibility of information for the user modelling module. The previous chapter discussed four sources of information that can be used in implicit user model acquisition: user behavior observed by the system, system behavior observable by the user, the system's domain model and the user model itself. Any user modelling module will have access to the user model itself, but the availability of other sources of information may be limited by the type of system in which the user modelling module must work.

The natural language interface provides the user modelling module with an easily accessible source of information about both user and system behavior. Usually the parser will produce some form of *meaning representation language* (MRL) that has a well-defined semantics, such as first

[2]See [Pollack 82] for an analysis of the types of interactions that occur in an advice-giving situation involving a human expert.

order predicate calculus.[3] This MRL provides an easily interpretable source of information about user behavior. With a natural language interface the user modelling module should also have access to the response generated by the system. This information may be in a form used by the system when composing the contents of the response, or might be obtained by feeding the natural language text generated by the system back through the parser to obtain MRL statements representing the system's behavior.

An advisory system also has a readily accessible source of domain information. Since the advisory system is based on an expert system, a user modelling module can access not only knowledge about the contents and structure of the domain, but also the reasoning rules used by the advisory system. Although it may not be the case for all advisory systems, this work assumes the domain knowledge is relatively robust. This seems reasonable since a natural language interface is fairly sophisticated and would benefit from a robust domain model as well. Although not obvious now, the availability of the reasoning knowledge of the advisory system (as opposed to just the structural knowledge of the domain) will be very important in enhancing the user model acquisition capabilities of the system.

Mixed Initiative

In a mixed initiative dialogue, either participant can establish a new topic or goal of the discussion. Following Grosz and Sidner [Grosz 86], either participant can establish a new discourse segment in a conversation. This characteristic of cooperative advisory systems is important to the user model acquisition process.

In a mixed initiative dialogue, the user is free to volunteer information at any time. *What* is volunteered and *when* can provide significant information about the beliefs held by the user. Further, in an advisory system it is acceptable for the system (advisor) to ask for further information from the user. The system is able to ask questions to obtain information it deems necessary, and even ask follow-up questions to clarify information the system is uncertain about. Thus a user modelling module could even generate questions to ask the user, perhaps to resolve an inconsistency detected in the user model.

A second advantage of mixed initiative interactions in an advisory system is the ability to recover from mistakes. Frequently, participants in a conversation make incorrect assumptions about their conversational partner. In a mixed initiative interaction, both participants share responsibility for ensuring the validity of the communication. This shared responsibility means that each participant can make assumptions about the other individual that seem plausible but for which there is insufficient evidence, knowing that if the other participant detects an error he or she will mention and correct the mistake. Thus a mixed initiative dialogue allows more freedom for an advisory system to make assumptions about the user, trusting that the user will attempt to correct any assumptions that are in error.

A third advantage of mixed initiative interactions in an advisory system is that users may sometimes make direct statements about their own beliefs. Thus a user might directly tell the system that s/he does not know much about a certain aspect of the domain. Such statements can be very helpful in modelling the user.

Cooperative

The fact that the user is cooperating with the system in the dialogue will prove important in acquiring information about the user. The user modelling module can assume that the user is

[3] At least, ideally the semantics are well defined. This assumption will be made for the remainder of this paper.

obeying the *cooperative principle* [Grice 75], in particular, the Gricean maxims of quantity, quality, relation and manner. The assumption of cooperativity places strong constraints on the interaction between user and system, constraints the user model can greatly benefit from.

Interesting and Useful

Although the type of interaction to be considered has been greatly restricted, cooperative advisory systems still constitute an important and interesting class of systems. Advisory systems are becoming more commonly used. Many of the expert systems now being developed attempt to capture the expertise of knowledgeable individuals. The system domains range from analysis of memory dumps taken following computer system crashes, to medical diagnosis, to industrial soup cooker repair. In each of these systems the goal is to create an advisory system with the expert's knowledge to aid other humans in dealing with the problems they face. Thus the class of systems to be considered is still a broad and important one.

User modelling can be very useful in cooperative advisory systems. Current advisory systems for the most part are not very cooperative. They require a certain level of knowledge on the part of the user, and frequently will not accept information volunteered by the user until the system explicitly asks for it. A user model can enhance the interaction capabilities of such systems by providing a source of information about the user's beliefs and goals in the current situation. This knowledge can be used to help tailor explanations, recommendations, and even the reasoning of the system.

Finally, although the range of interaction has been restricted, the problem of user model acquisition is still quite challenging. In fact, user modelling in such a domain has seldom been attempted,[4] and never has implicit model acquisition in such a domain been emphasized. One reason for the lack of work in this area has been the general impression that it is too complex. Certainly the requirements for communication in natural language in such a context are more difficult than, say, question answering systems. Consequently user modelling in the domain seemed more difficult as well. Despite the greater complexity of the domain however, the form of interaction can provide more information about the user, information that a user modelling module can acquire implicitly.

Good Paradigm for Study

A last advantage for studying cooperative advisory systems is that a good paradigm for study exists. People frequently seek advice from others. These interactions serve as a basis for studying the modelling that a human expert does while responding to a request for advice. The next chapter contains a collection of rules that a user modelling module can use in acquiring information about a user from a cooperative advisory interaction. These rules were obtained by analyzing transcripts of human advice giving.

4.3 Long Term Models

Since the focus of this work is on implicit model acquisition techniques, the types of models to be considering are individual models of the user. Furthermore, the kind of information of interest is information that remains true about the user for some period of time. Thus the goal is to build models that remain useful over many interactions between the user and the system.

[4]The primary example of user modelling in an advisory domain is GUIDON [Clancey 82,London 82], which modelled the medical reasoning knowledge of students learning to diagnose forms of meningitis.

This long term nature of the model minimizes the importance of some types of model information. In particular, system beliefs about user goals and plans tends to be relatively short term information.[5] For this work, the acquisition of knowledge for the user model will focus on beliefs held by the user—in particular, beliefs the user holds about the domain of the system. This is appropriate for an advisory system, since much of what the system advisor would want to know about the user is what the user knows about the domain.

[5]Actually, some goals and plans may persist for long periods of time; these will not be consisdered here.

Chapter 5

User Model Acquisition in Cooperative Advisory Systems

This chapter presents the central contributions of the paper. The previous chapters have included criticisms of explicit model acquisition techniques, and arguments for acquiring user models implicitly. Here a set of implicit acquisition rules for user modelling is presented. The rules as stated cannot be directly implemented in a user modelling system, but will serve as a basis for such an implementation. Thus, this chapter presents a foundation on which a domain-general user modelling module can be built. In sections 5.1 and 5.2 the nature of these rules and how they were obtained is discussed, while in sections 5.3–5.6 the rules are presented.

5.1 The Transcripts

The acquisition rules for the most part have been developed by examining transcripts of a large set of interactions (close to 100) between individuals and an expert. These transcripts were made from recordings of a radio talk show entitled "Harry Gross: Speaking about Your Money" broadcast by radio station WCAU in Philadelphia, on February 1–5, 1982.[1] The examples relating to investments in the remainder of this chapter are drawn from these transcripts.

Several similarities between the radio program and computer advisory systems make study of these transcripts useful. In both cases the medium of communication is strongly restricted. Human interaction with a computer expert is generally limited to a keyboard and display screen. On the radio program the communication is verbal, primarily spoken language between the two participants. This means a transcript can capture most of the information communicated between participants. Thus the transcripts from the radio program are quite similar to the type of interaction between human and computer.

A second advantage of studying the radio program transcripts is that the callers vary quite a bit in their knowledge of the domain of their questions. For example, the expert must deal with people who know quite a lot about investing, as well as people who have almost no knowledge of the subject. Thus the expert cannot rely on some common model for all callers, but instead must actively model each new caller in order to deal with them effectively.

A final advantage of studying the radio program transcripts is that the expert generally has not dealt with the caller before. Thus the expert must construct a model of each caller during the course of the conversation, based only on the information available in the conversation. The model

[1]These transcripts were also used as a basis for the findings in [Pollack 82]. Many thanks to Martha Pollack and Julia Hirschberg who recorded and transcribed the show.

of the caller that the expert develops is therefore solely a result of the interaction recorded in the transcripts.

The fact that the transcripts are of a radio program does not detract from the information gained about modelling. It might seem that the radio show format would be too artificial to demonstrate normal human behavior. This is partly true, but does not affect the modelling aspects on the part of the expert. The main effect the radio show format seems to have on the interactions is a tendency for the expert to omit detailed explanations that would be boring to the audience as a whole. For example, the expert frequently recommends investment in a money market fund. Many of the callers are unfamiliar with money market funds, yet the expert will often omit a detailed explanation, because he has already made a similar explanation earlier in the show. This brevity of explanation does not affect the actual processes used to model the knowledge of the caller.

Although the rules to be presented here were developed by analyzing the radio program transcripts, that does not mean that these rules, and only these rules, were used by the expert. In fact, there is ample evidence that the expert makes use of certain stereotypes, such as for "senior citizens" or "widows," in reasoning about a caller's situation and knowledge. On the other hand, the rules to be presented *will* account for most of the information about the caller used by the expert in the course of the conversation.

5.2 Effectiveness Versus Efficiency

It is useful to draw a distinction between *effectiveness* and *efficiency* in user modelling. The expert in the transcripts is experienced in advising a variety of individuals about investments. Through experience he has formed stereotypes or rules of thumb such as "Widows usually don't know much about securities other than those available from a bank." These stereotypes enable the expert to very quickly form a model of a new caller, based on his experience with previous individuals. Thus experience enables the expert to be a very *efficient* modeller, in that he can quickly develop a robust model of the caller without a great deal of reasoning.

A general user modelling module is in a situation similar to a person trained to repair computers, but who has never worked in the field. Such a person knows circuit theory and how the computer works, but does not have the practical knowledge of which components are likely to fail that is gained through experience. Likewise, a general user modelling module has available the domain knowledge of the application, but does not know the correlations between classes of users and knowledge of concepts or rules in the domain model. Thus a general user modelling module cannot make the quick, efficient leaps that are possible with a stereotype-based user model.

Although the efficiency of a general user modelling module may be limited, it can still be *effective*. In fact, the rules to be presented in this chapter, taken together, are sufficient to provide most of the user modelling necessary to support the cooperative and helpful behavior of the expert in the conversations analyzed. The user modelling module may perform many more inferences to arrive at a model of the user than a stereotype-based user modelling system might, but the rules are domain independent. Thus the user modelling module can maintain its effectiveness when the underlying domain is changed, something a stereotype-based user modelling system cannot do.

The beliefs about the user obtained by applying these rules are not absolute. As with any user model, the information believed about the user may be wrong. In addition, it is possible for the rules to draw conflicting conclusions about the user's knowledge. These potential conflicts will necessitate some means for arbitrating among sets of beliefs. The rules to be presented here were written with no assumptions about the arbitration technique. It is convenient to think of the modelling rules as being *default rules* [Reiter 80], but other techniques could be applied as well.

For example, evidential reasoning could easily be accommodated by adding evidence weights to the rules, if this were desired.

Exceptions to the acquisition rules exist. For each of the rules, situations in which the rule makes an incorrect conclusion about the user's beliefs can be constructed. In fact, such exceptions are frequently easy to find. The important point is that such situations *are* exceptions. The goal of this work is to present *reasonable* rules, rules that draw correct conclusions most of the time. Furthermore, when an incorrect conclusion is drawn by the user modelling module, the error should be a reasonable mistake. Often people develop beliefs about others that are incorrect. Thus this model acquisition technique reflects the imprecision humans experience in modelling individuals.

The rules to be presented can be loosely divided into three categories: communicative rules, model-based rules and human behavior rules. Communicative rules are based on aspects of the communication between user and system. Thus rules that infer portions of the user model directly from a user's statement, and rules based on the assumption that a user is cooperating with the system are communicative rules. Model-based rules are rules that depend on certain relations between information in the user model, or on certain characteristics of the domain model. Although domain related, these rules are independent of the contents of the domain. Human behavior rules depend on the way people behave, such as how they reason or take action. Note that these categories do not form firm divisions between the various classes of rules. There is a lot of overlap, for example, between human behavior rules and communicative rules.

The communicative, model-based and human behavior rules are presented in the next three sections. In this discussion, only three of the sources of information for a user model will be used: the statements made by the user, the current user model and the domain model. The statements made by the system will be omitted. Statements made by the system have little impact on most of the implicit acquisition rules. The final section contains a brief discussion of what information can be obtained for the user model from system statements, and how previously defined rules can be used to handle these situations.

5.3 Communicative Rules

The communicative rules can be subdivided into two classes: direct inference rules and implicature rules. The direct inference rules draw conclusions based solely on the statement made by the user. The implicature rules are inspired by Grice's maxims, and assume that the user is behaving in a cooperative manner.

5.3.1 Direct Inference Rules

There are two direct inference rules. The direct statement rule updates the user model with the content of statements made by the user. The presupposition rule adds information that is presupposed by user statements.

Direct Statement Rule

The direct statement rule says:

Rule 1 *The user modelling module can assume that a statement made by the user is believed by him.*[2]

[2]For future reference, the implicit acquisition rules presented in this chapter are summarized in appendix A.

$$\begin{array}{ll}\exists x_1 & (\text{investment}(x_1) \wedge \\ & \exists x_2(\text{investor}(x_1, x_2) \wedge x_2 = \text{user}) \wedge \\ & \exists x_3(\text{instrument}(x_1, x_3) \wedge \text{moneymarket}(x_3)) \wedge \\ & \exists x_4(\text{amount}(x_1, x_4) \wedge \text{dollar}(x_4) \wedge x_4 = 40,000))\end{array}$$

Figure 5.1: MRL for "I have $40,000 in money market"

Many beliefs of the user can be asserted to the user model by this rule. In fact, there is a straightforward method for extracting individual beliefs that can be attributed to the user from the statements made by the user.

To demonstrate how the direct statement rule works, consider the statement:

I have $40,000 in money market.

Figure 5.1 illustrates the MRL that a parser could produce for this statement.[3] This logical expression reflects the fact that the user has made a statement about an investment. Investments have investors, instruments and amounts. In this case the investor is the user, the instrument is money market (securities) and the amount is $40,000. From this logical expression 15 component sub-expressions can be derived, corresponding to individual beliefs held by the user. Each expression, and an interpretation in English, is given in figure 5.2. These component beliefs derived from the user's statement may now be used to update the user model. Assuming a KL-ONE [Brachman 85a] type of knowledge representation for the beliefs the user holds about objects and concepts, the taxonomy in figure 5.3 can be produced.[4]

Do not be misled by the seeming richness of the representation produced. Terms like "investment" and "instrument" have been used as labels for the concepts that the user knows. These labels are really just convenient names to use when referring to particular concepts. The notion of "instrument" might imply that the user has a fairly deep knowledge of the domain and a firm understanding of various aspects of investments. This need not be the case. Rather, the user may simply have the notion of "having put money" in a particular place. An expert would interpret this act as having invested in a certain financial instrument, but would not assume the user also held knowledge about investments or financial instruments in general. The user model then, should be interpreted on the basis of the structure of concepts and roles, and their relationships in the knowledge representation hierarchy. This point is also made by Brachman in describing how to interpret representations of knowledge in KL-ONE.

There is an interplay between the natural language understanding (NLU) component of a system and the user model. The NLU component must use knowledge in the system to interpret what the user says. If only domain knowledge is used, the MRL produced may not be representative of the beliefs of the user. A better approach might have the NLU component use knowledge from the user model, as well as domain knowledge, when interpreting user statements.

The direct statement rule is not new. Other systems have produced output from the parser that can be directly applied to the user model. Examples include HAM-ANS [Hoeppner 83], the

[3]For simplicity, assume the MRL is first-order predicate calculus.

[4]The steps described here have already been implemented as a Prolog program. This program will take as input a logical expression such as the one in figure 5.1 and produce the list of individual clauses in figure 5.2. The program then uses these clauses to generate assertions to a NIKL [Moser 83] knowledge base, producing the structure in figure 5.3. The program runs on a Symbolics Lisp Machine.

$\exists x_2(x_2 = \text{user})$	"User" is an entity	(5.1)
$\exists x_4(x_4 = 40,000)$	40,000 is an entity	(5.2)
$\exists x_1(\text{investment}(x_1))$	Investment is a concept	(5.3)
$\exists x_3(\text{moneymarket}(x_3))$	Moneymarket is a concept	(5.4)
$\exists x_4(\text{dollar}(x_4))$	Dollar is a concept	(5.5)
$\exists x_1, x_2(\text{investor}(x_1, x_2))$	Investor is an attribute	(5.6)
$\exists x_1, x_3(\text{instrument}(x_1, x_3))$	Instrument is an attribute	(5.7)
$\exists x_1, x_4(\text{amount}(x_1, x_4))$	Amount is an attribute	(5.8)
$\exists x_1, x_2(\text{investment}(x_1) \wedge \text{investor}(x_1, x_2))$	Investments have investors	(5.9)
$\exists x_1, x_3(\text{investment}(x_1) \wedge \text{instrument}(x_1, x_3))$	Investments have instruments	(5.10)
$\exists x_1, x_4(\text{investment}(x_1) \wedge \text{amount}(x_1, x_4))$	Investments have amounts	(5.11)
$\exists x_1, x_3(\text{instrument}(x_1, x_3) \wedge \text{moneymarket}(x_3))$	Moneymarket can be an instrument	(5.12)
$\exists x_1, x_4(\text{amount}(x_1, x_4) \wedge \text{dollar}(x_4))$	Dollars can be an amount	(5.13)
$\exists x_1, x_3(\text{investment}(x_1) \wedge \text{instrument}(x_1, x_3) \wedge \text{moneymarket}(x_3))$	Moneymarket can be the instrument of an investment	(5.14)
$\exists x_1, x_4(\text{investment}(x_1) \wedge \text{amount}(x_1, x_4) \wedge \text{dollar}(x_4))$	Dollars can be an amount of an investment	(5.15)

Figure 5.2: Decomposition of "I have $40,000 in money market"

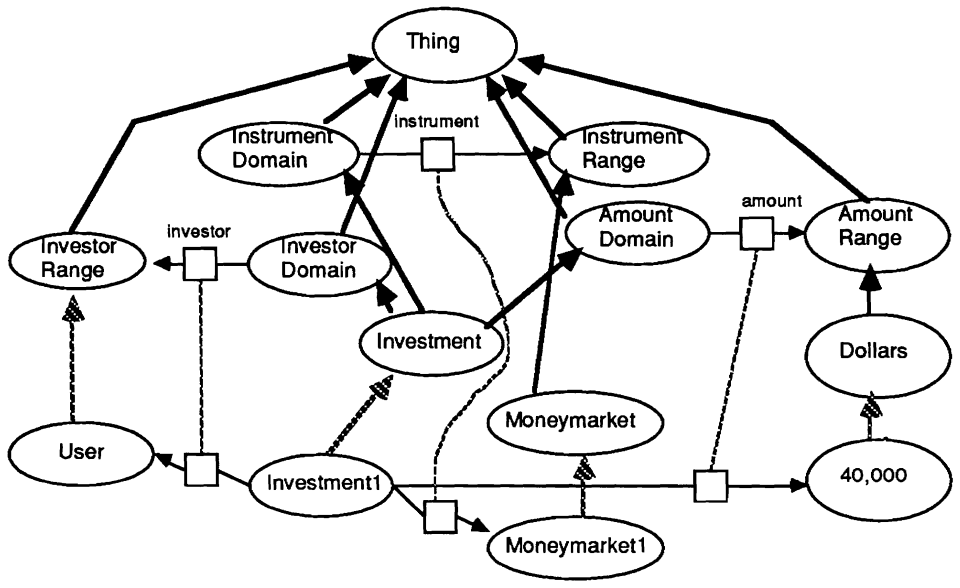

Figure 5.3: KL-ONE representation of concepts derived from "I have $40,000 in money market"

UNIX Consultant [Wilensky 86] and VIE-LANG [Kobsa 84]. Some systems, such as the UNIX Consultant, directly generate a knowledge representation structure for the user's statement. This knowledge representation structure can then be directly asserted to the user model.

The direct statement rule provides some basic information about what concepts the user is familiar with. One advantage of this rule is that the results are usually quite certain, since statements made by the user usually reflect what is believed. On the other hand, the direct statement rule alone only gives information about concepts specifically mentioned in the interaction. This leaves vast amounts of information about the user untouched.

The Presupposition Rule

A presupposition of a statement is something that must be true in order for the statement to make sense. Formally, P presupposes Q if and only if $(P \Rightarrow Q) \vee (\neg P \Rightarrow Q)$. Presuppositions can be especially important in dealing with questions from the user. For example, the question

Which students received A's in CIS577 last fall?

presupposes, among other things, that CIS577 was taught during the fall, and that an "A" was a potential grade for the course.

Presuppositions have been studied for some time. Kaplan's COOP system [Kaplan 82] focused on identifying incorrect presuppositions in user queries to a database system in order to provide more cooperative negative responses. For example, if CIS577 was not offered last fall, a valid answer to the preceeding question would be "none." This would reaffirm the user's incorrect belief that CIS577 had been offered. A more cooperative response would be "None, CIS577 was not offered last fall."

Kobsa has addressed computing presuppositions in the context of modelling the user in the VIE-LANG system [Kobsa 84]. For example, from the question

To whom does Peter give the book?

the following assumptions can be made.

1. The user believes Peter and the book exist, and that Peter gives the book to someone unknown to the user.

2. The user believes that the system believes that Peter gives the book to someone. The user also believes that the system knows the recipient.

3. The user wants to be in a situation in which both the user and the system know who Peter gave the book to.

With these examples, the presupposition rule can be stated as follows.

Rule 2 *The user modelling module can assume that presuppositions of statements made by the user are believed by him.*

5.3.2 Implicature Rules

The implicature rules are based on Grice's *cooperative principle.* Grice has observed that humans obey certain conventions when communicating with each other [Grice 75]. One such convention is that they strive to be cooperative during conversations with each other. He has stated this cooperative principle as follows:

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged. [Grice 75, p 67]

The implicature rules are based on the assumption that the user of the system is observing this cooperative principle.

The term "implicate" holds a specialized meaning in Grice's usage. Information that is implicated by a statement cannot be directly inferred from that statement. Using an example given by Grice, suppose A and B are talking about a mutual friend C who is now working at a bank. A asks B how C is doing in his job and B replies "Quite well, he likes his colleagues, and he hasn't been to prison yet." The statement seems to imply something beyond the simple fact that C has not been to prison yet, perhaps that A is trying to suggest that C will be going to prison because he is stealing money from the bank, or because his colleagues are crooked. The information implicated by a statement is thus beyond the logical content of the statement itself, and hence cannot be directly inferred from the statement. Being able to recognize what is implicated by a user's statements could furnish a great deal of additional information about the user.

Grice proposes four categories of maxims that cooperative people observe when making an utterance. These categories are the maxims of quantity, quality, relation and manner. Grice has stated the maxims as rules that people tend to follow when making an utterance. These maxims are important for implicit user modelling because they provide guidelines for the expected behavior of the user. In the following paragraphs each of the maxims is stated, along with a paraphrase reflecting the user modelling perspective.

Quantity The maxims of quantity are: "make your contribution as informative as is required," and "do not make your contribution more informative than is required." From the standpoint of a user modelling module, this maxim can be paraphrased as follows.

> *The user believes that his statement provides sufficient information to accomplish his current goal or goals, and does not include information extraneous to accomplishing those current goal or goals.*

Assume that the "current goal" of the user can be easily recognized. In an advisory system, this will generally be true. Often the system will have asked the user a question, so that the current goal held by the user should be to provide enough information to answer the question posed by the system. Other goals might be higher-order goals such as giving or receiving advice about investments, or making a judgment about a particular aspect of the overall goal.

Quality The maxims of quality are: "Do not say that which you believe to be false" and "Do not say that for which you lack adequate evidence." From the viewpoint of a system modelling the user, the maxim of quality can be quite simply paraphrased:

> *The user believes his statement is true.*

The maxim of quality has already been assumed by the direct inference rules. These rules depend on the assumption that the user believes what he says.

Relation The maxim of relation is simply "be relevant." From the viewpoint of the speaker this maxim is the least specific. It does not provide any means for identifying how to be relevant, or even determining what information is relevant to say. Fortunately, from the standpoint of a user modelling module for an advisory system, the maxim of relation is less problematic. In this context, the maxim can be paraphrased to say:

The user believes the information he has provided is relevant to his current goal or goals.

In an advisory system, the user's goals will always center on obtaining advise about a particular problem or situation. The knowledge base for the expert advisor will include rules that reason about various facts that lead to conclusions used in responding to the user. The information the user believes is relevant is thus information the user believes will be used in the advisor's reasoning process.

Manner The maxim of manner pertains to how information is communicated. Grice lists several rules that a speaker should observe in order to abide by the cooperative principle, including:

- Avoid obscurity of expression.
- Avoid ambiguity.
- Be brief.
- Be orderly.

From the standpoint of user modelling, the maxim of manner is the least interesting, since it primarily affects the form of the utterance, not the contents. However, some use may still be made of this maxim. The maxim of manner can be paraphrased as follows:

The user believes his statement is clear, concise and well ordered.

This maxim will be of use to a user modelling module when the statement made by the user in fact violates the maxim. For example, if the statement made by the user is ambiguous, it indicates a lack of knowledge on the part of the user.

The paraphrases of Grice's maxims provide a basis for implicit user model acquisition rules that go beyond the direct inference rules discussed earlier. In particular, the maxims sometimes enable the user modelling module to identify reasoning the user has performed, as well as information the user *does not* know. This information can be obtained for two reasons. First, the cooperative principle provides a basis for establishing certain expectations about the user's behavior. Secondly, the domain knowledge base supplements these expectations with specific domain knowledge the user may be expected to have. Taken together, a powerful form of differential modelling can be achieved.

Grice's maxims inspire three implicature rules, based on the maxims of relation, quantity and manner, but the correspondence is not total. Each rule relies to some degree on the other maxims as well.

Relevancy Rule

The relevancy rule is inspired by the paraphrase of the maxim of relation: the user believes that what he says is relevant to his current goal. The rule says:

Rule 3 *If the user says P, the user modelling module can assume that the user believes that P, in its entirety, is used in reasoning about the user's current goal or goals.*

The use of the relevancy rule can be illustrated with an example. The following statement is made at the beginning of a call to the expert.

(1) C. I have an $18,000 golden passbook account which pays me $7\frac{1}{4}$%

(2) which is going to mature in April of this year.

(3) Now I'd like to know what to do about it.

Here, although somewhat unclear, the caller wants advice from the expert on how to invest the \$18,000 once it becomes available. The caller believes that the amount of money, the due date and the current interest rate are important factors in making a decision about how to invest the money so she states them with the initial request for advice. Thus the expert, in modelling the caller, can conclude that the amount, interest rate and due date of the golden passbook account are believed by the caller to be used in the reasoning about how to reinvest the money.

In addition to claiming that the user believes what is said is relevant, the relevancy rule states that the user believes that *everything* in the statement is relevant. This can be illustrated with another example.

(4) C. I just retired December first,

(5) and in addition to my pension and social security

(6) I have a supplemental annuity

(7) which I contributed to while I was employed

(8) from the state of New Jersey mutual fund.

(9) I'm entitled to a lump sum settlement which would be between \$16,800 and \$17,800

(10) or a lesser life annuity

(11) and the choices of the annuity would be \$125.45 per month.

(12) That would be the maximum with no beneficiaries.

(13) E. You can stop right there, take your money.

In this example the caller provides a large amount of information in stating her problem. The expert recognizes that the only information relevant to the question of how to take her supplemental annuity is the value of the annuity if taken as a lump sum versus the monthly payments that could be received. Once the expert has this information (and has identified the goal the caller has) he interrupts and provides the answer. Meanwhile, in modelling the caller the expert can conclude that she has little knowledge of the reasoning involved in making the decision. She feels that all the information she has listed may be relevant to the reasoning process, while in fact it is not. Thus the relevancy rule can be used to acquire information about incorrect reasoning a user may perform.

Sufficiency Rule

The sufficiency rule is inspired by the maxim of quantity. The system can reason as follows: if the user were completely knowledgeable about the domain, he would provide information sufficient for the system to satisfy the user's goal. Suppose what the user says turns out to be insufficient? In this case the user must lack some knowledge that the system has. A user may have three types of knowledge about entities in the domain knowledge base:

1. The user may *know of* the entity. For example, from the statement "I have $40,000 in money market" the direct statement rule concludes that the user knows of the concept "money market."
2. The user may *know the relevance* of the entity. This means that the user knows the entity is used in reasoning about the current goal.
3. The user may *know the value of* a property. Concepts that represent actual entities in the world have roles that take on real values. Thus 40,000 is the value of the amount role for the concept "investment1" in figure 5.3.

When the user is cooperative, yet omits a piece of information that the system knows is relevant, it is due to a lack of knowledge of one of these three types.

The sufficiency rule is stated as follows:

Rule 4 *If the user omits a relevant piece of information from a statement, then either the user does not know of that piece of information, does not know whether that information is relevant to his current goal or goals, or does not know the value for the piece of information.*

Once again an example will illustrate this rule.

(14) C. I've got $2250 to invest right now in an 18 month certificate

(15) and I don't know whether to go

(16) the variable rate or the fixed rate now or the fixed rate later.

(17) E. Let me ask you a couple of questions.

(18) C. OK.

(19) E. Have you any money invested now?

(20) C. Yes, I do.

(21) E. In what?

(22) C. I've got $5000 in a money market fund.

(23) E. Have you anything in certificates or anything else?

(24) C. I've got three stocks.

(25) E. Three separate stocks?

(26) C. Yes sir.

In this conversation, the caller believes his initial statement of the problem is sufficient for the expert to make a decision. Instead, the expert realizes a lot more information about the caller's investments are needed. The expert proceeds to ask questions to obtain this information. Even then, the caller provides minimal answers because he does not know what additional information is relevant until the expert specifically asks for it. In this case it seems obvious that the user knows the additional information, he just does not realize it is relevant.

In using the sufficiency rule, the user modelling module must be able to "turn around" the reasoning rules in the domain model, in order to identify properties that are relevant. This collection of relevant properties creates an expectation of the information the user should provide. Information in the set of expectations that is not provided thus must be information the user lacks knowledge of.

The sufficiency rule might be strengthened further. If the user is being fully cooperative he will try to be as helpful as possible. Suppose the user knows a piece of information, but does not know its value. For example, the user might know that the due date of a money market certificate is relevant information, but not know the actual due date. A truly cooperative user would tell the system that he does not know the due date. Thus the sufficiency rule might be limited to conclude that either the user does not know of the information, or does not know that it is relevant. Furthermore, if the user does not know of the information, he certainly cannot believe it is relevant, so the sufficiency rule could make a definite conclusion in this case.

Although the strengthened sufficiency rule seems attractive, that level of cooperation by the user does not seem likely. People are reluctant to display their ignorance. Thus, when they don't know something, they avoid mentioning it, even when they believe it is relevant.

Ambiguity Rule

The ambiguity rule is based on the maxim of manner. A portion of the paraphrased maxim states that the user believes what he says is unambiguous. Suppose the system finds the user's statement to be ambiguous. In this case, the user must lack knowledge of interpretations of the statement other than the one intended. Thus the ambiguity rule provides the user model with information about objects, concepts and rules in the domain that the user does not know. The ambiguity rule can be stated as follows.

Rule 5 *If the user makes a statement that the system finds ambiguous in the current context, then the user lacks knowledge of one or more of the alternative meanings for his statement.*

The ambiguity rule can be illustrated in the following example.

(27) C. We have $40,000 in money market,

(28) one is coming due, a $10,000 money market.

In this example, the first line could be interpreted to mean that they have a total of $40,000 invested in money market instruments. However, the use of "money market" in the second line indicates that the term is not being used in that sense. In this conversation the expert did not seek immediate clarification of what was meant by "money market." Rather, by letting the conversation progress it became obvious that the caller was referring to money market certificates that are issued by banks, and had no knowledge of other money market securities.

The most common ambiguities seem to be related to the use of individual terms. In these cases the speaker will tend to use a term that is more general than the appropriate term. This confusion usually arises because the speaker is not aware of some of the specializations of the more general term that is used. This is the case in the example, where the caller thought the term "money market" was a synonym for the term "money market certificate."

5.4 Model-Based Rules

While the communication rules are triggered by statements made by the user, the model-based rules depend on features of both the domain model and the current user model. The model-based rules look at what the system currently believes the user knows in order to find plausible extensions to the user model. Thus the model-based rules compare the user model to the domain knowledge, in order to augment the user model with domain knowledge.

The model-based rules (in fact any rule that uses domain knowledge) assume certain constraints on the knowledge representation used. This work assumes that domain knowledge is of two types: knowledge about concepts, properties and their relationships (conceptual knowledge); and reasoning knowledge used in the advisory process. The examples of conceptual knowledge in this paper use a KL-ONE type of representation, but the essential features are the association of properties with concepts, a subsumption hierarchy for concepts, and property inheritance in that hierarchy. Thus a frame-based representation could be used as well. The reasoning knowledge is assumed to use a rule-based representation. Here it is important to identify the properties used in reasoning, and the dependencies between them.

There are eight model-based rules that will be discussed in the following section: the action rule, the consequence rule, the concept generalization rule, the consequential generalization rule, the distribution rule and three entailment rules.

5.4.1 Action Rule

The action rule relates an action to its preconditions and postconditions. Sometimes a user's statement will indicate that he knows of an action. Consider, for example, the statement:

(1) C. We just rolled over two money market certificates.

Here the user makes explicit reference to the action of "rolling over" a money market certificate.[5] The direct statement rule would assert that the user knows a concept corresponding to the roll over action, but nothing else about that action. It is reasonable to assume that if a user knows about an action, then he knows about conditions that must be true for that action to be performed, as well as the consequences of the action. The rule can be stated as follows:

Rule 6 *If the user model includes the belief that a user knows an action, then the user modelling module can attribute to the user knowledge of the preconditions and postconditions of that action.*

Consider the statement

(2) C. I was thinking of buying a treasury note.

The caller has expressed knowledge of the action of buying a treasury note. The action rule will assert that the user therefore knows that to buy a treasury note requires that the agent possess a certain amount of money (equal to the face value of the denomination of treasury note being bought). Further, the action rule will assert that the user knows that the result of buying a treasury note is that the agent possesses a piece of paper with a face value equal to the value of the denomination purchased.

5.4.2 Consequence Rule

The consequence rule might be considered the reverse of the action rule, relating the results of an action to the action itself. The consequence rule has two parts, affecting not only the user model, but the domain model as well. If the user believes in the consequence of an action, it is plausible to assume that the user believes that the action has occurred. The consequence rule also affects the

[5]To roll over an investment is to remove money from an investment in one security and immediately reinvest that money in another security, so as to avoid paying taxes.

domain model by indicating that the action has occurred. If the system believes in the consequence of an action, then the domain model should be updated to believe that the action has occurred as well. Such actions may subsequently trigger the agent rule (one of the human behavior rules to be discussed later) to add additional information about the beliefs of the user.

The two parts of the consequence rule can be stated as follows.

Rule 7

i If the user believes a fact that can only result from one of a small set of actions, then the user believes that one or more of those actions has occurred.

ii If the system knows a fact that can only result from one of a small set of actions, then the system believes that one or more of those actions has occurred.

Determining the size of a "small set" is a matter of preference. If the set of possible actions that caused an event were very large, asserting that the user knew one of these actions would not provide much information to the user model. If the set consists of a very few items, other acquisition rules might be used to eliminate some possibilities, or to strengthen justifications for others. The ideal "small set" is a set of one, since then the action causing the event can be completely identified. Even when the set consists of more than one action, there may be common subparts to the actions that can be believed with greater certainty.

An example of the use of the consequence rule can be seen in the following statement.

(3) C. I have $2300 in a passbook account.

The direct statement rule will assert that the user believes that she has $2300 in a passbook account. The presence of this money in the account is dependent on the fact that someone must have deposited money into the account. In this case, the action of opening an account is a necessary prerequisite for depositing money in the account, so the consequence rule will conclude that an account was opened, and that money was deposited to the account.

5.4.3 Concept Generalization Rule

The concept generalization rule seeks to fill out the concept hierarchy of the user model with information from the domain model. Although the user model need not mirror the domain model, if the user model develops several beliefs that seem to match those of the domain model, the concept generalization rule will attempt to augment the user model with pieces of the domain model. The formal statement of the concept generalization rule is:

Rule 8 *If the user model indicates that the user knows several concepts that are specializations of a common, more general concept in the domain model, the user modelling module may conclude that the user knows the more general concept, and the subsumption relationship between the more general concept and the more specialized concepts as well.*[6]

For example, suppose the user has indicated knowledge of "red," "blue," "green" and "yellow." The concept generalization rule would use the knowledge in the domain model to assert that the user knows the concept "color." Further, the rule would assert that the user knows that red, blue, green and yellow are kinds of colors.

[6]I am assuming a KL-ONE type of knowledge representation. A concept A subsumes a concept B if and only if every instance of B is an instance of A. In KL-ONE, concepts are placed in a lattice, with directed links representing the subsumption relations between them.

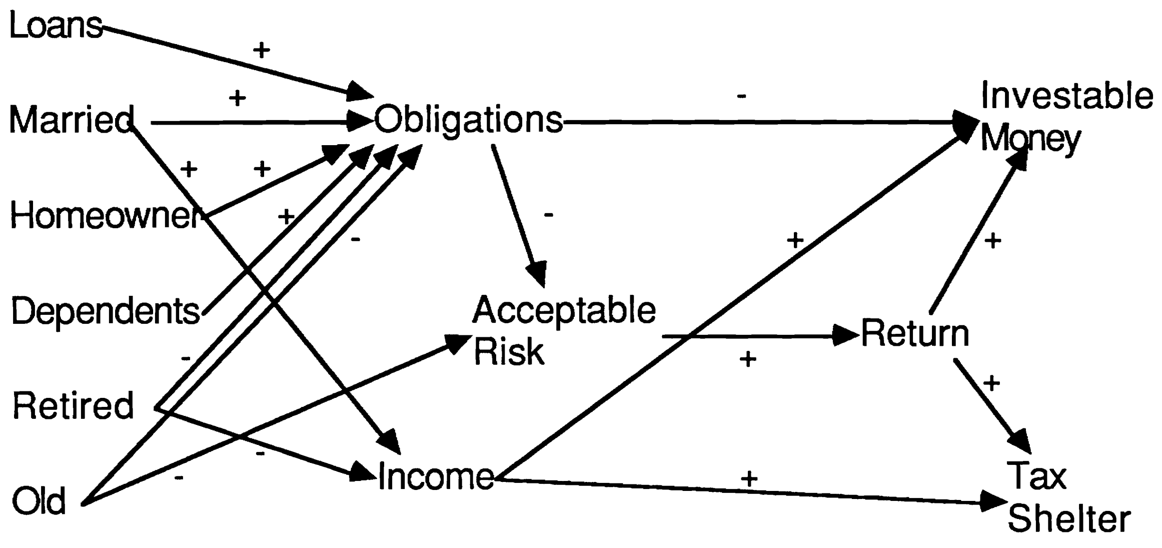

Figure 5.4: A sample rule graph. An arc from node A to node B indicates that A supports B. '+' indicates positive support, while '-' indicates negative support.

The concept generalization rule is triggered when the user knows *several* concepts that have a common generalization. The actual number of concepts is probably quite small, perhaps 3 or 4. People tend to quickly draw generalizations, so if the user indicates knowledge of 3 or 4 concepts, it is reasonable to assume that he also knows of a more general concept that subsumes them.

5.4.4 Consequential Generalization Rule

The consequential generalization rule is similar to the concept generalization rule, only it operates on rule-based knowledge. The concept generalization rule works by using the subsumption hierarchy to discover immediate subsumers of concepts believed to be known by the user. Likewise, the consequential generalization rule uses the graph structure of rules to find immediate consequences the user is likely to know.

Before presenting the consequential generalization rule, an explanation of the rule structure is necessary. The rule base of an expert system can be viewed as a graph structure. The nodes of the graph are properties used in reasoning by the expert system. The links represent the dependencies between properties that are induced by the rules. For example, an expert system that gives advice about investments would have rules relating properties of the user such as age, income and number of dependents to the amount of risk they can accept. Acceptable risk then determines the classes of investment vehicles that should be considered in a particular situation. This example is illustrated in figure 5.4.

The consequential generalization rule reasons on the dependency graph of properties for the expert system. If the user believes that a certain set of properties is relevant, then a property affected by this set, and the links from the set to this property may be known by the user. The consequential generalization rule is:

Rule 9 *If the user model indicates that the user believes that several properties having a common consequence are relevant, then the user modelling module may assert that the user knows of the common consequent, believes that it is relevant, and knows of the links between the set of properties and the common consequent.*

Consider the following conversation.

(4) C. I would like to get your advice today on what I should do
(5) with some of my assets.
(6) E. Sure.
(7) C. OK. I'm 27, recently married.
(8) My wife makes $20,000 a year.
(9) I make about 23 or 24 thousand depending on commissions and bonuses.
(10) We have some certificates.
(11) We have an All Savers certificate that's due next October.
⋮

In this conversation the caller happens to be fairly knowledgeable about investments. Thus he supplies information pertinent to determining how much risk he and his wife can accept, what their total income is, and so on. The consequential generalization rule will allow a user modelling module to conclude that this person knows about properties such as risk and investable income, and perhaps others as the conversation progresses.

Although the consequential generalization rule can indicate that the user recognizes dependencies between properties, it cannot indicate what kind of dependency exists. Dependency links may supply either positive or negative support for a property. For example, in figure 5.4, both "retired" and "dependents" influence the financial obligations a user has, but in opposite ways. If the user has dependents, his obligations tend to be greater, while a retired user would tend to have fewer obligations. If the user provides information about retirement status and number of dependents (as well as other properties that influence financial obligations), the consequential generalization rule will be able to conclude that the user knows of the "obligations" concept, and of the dependency links connecting "dependents" and "retired" to "obligations." However, the rule is not capable asserting whether the user knows that being retired tends to reduce obligations—only that the user knows that retirement status influences obligations.

5.4.5 Distribution Rule

The distribution rule depends on the structure of the knowledge representation, but in a way different from the generalization rules. The distribution rule relies on the current user model, the domain model, and the user's statements to augment the user model. In some situations the system will expect a response from the user. The domain model will specify the range of possible responses. If the user's response is limited to some subset of this range, it is evidence that the user lacks knowledge of the other possible items for the response. The distribution rule is stated as follows.

Rule 10 *Given a situation in which the user is to provide a list of items from a class for a response, if the user's response consists only of a particular subset of those items, then the user modelling module can assert that the user lacks knowledge of the other items in the class.*

Again an example will illustrate the rule.

(12) C. We have $40,000 in money market.

(13) One is coming due, a $10,000 money market.

(14) We have $2300 each in a savings account at 6%

(15) and then we have $1000 each in a credit union at 7%.

(16) What I'd like to know is whether you think this is good savings,

(17) or how you think we might do better,

(18) maybe on a treasury bill?

The expert's goal, which is determined at the end of the caller's statement, is to evaluate the caller's investments. The expert thus expects the caller to list her investments, which should be in the range of investment securities. The caller has provided this list, but the items listed all fall under a small subcategorization of securities, fixed income debt securities. Further, all the investments currently held are obtainable from banks, credit unions or savings and loan institutions. The distribution rule would conclude from this example that the user does not know much about non-fixed income securities, investments such as stock, commodities or options. In fact, she may not have knowledge of securities beyond those available at bank or bank-like institutions.

The effectiveness of the distribution rule depends on three restrictions on the selection of items. First, the selection should not call for a limited set of items. For example, if the user is asked to choose only three from a class of items, the fact that some items were omitted cannot be attributed to lack of knowledge, it might be due to the selection limitation. Likewise, the items in the selection class should not be interdependent. If the choice of item A precludes choosing B, then B's omission may be due either to lack of knowledge or to the fact the A was chosen. Finally, the selection process should not be externally influenced. For example, if personal preferences influence the selection, items may be omitted because they are not preferred, not because of lack of knowledge.

The distribution rule can be a powerful technique for identifying large areas in which the user lacks knowledge. However, it must be used with caution. The restrictions on the rule are significant and must be observed. If not, the user model could hold drastically incorrect beliefs about the user.

5.4.6 Entailment Rules

The three entailment rules make assertions to the user model based on logical inferences the user is expected to perform. They are included with the model-based rules because their conclusions are implicit in the representational structures. The entailment rules consist of the transitive subsumption rule, the inheritance rule and the transitive reasoning rule.

Transitive Subsumption Rule

The transitive subsumption rule states:

Rule 11 *If the user believes A subsumes B, and that B subsumes C, then the user believes that A subsumes C.*

For example, if the user believes that treasury bonds are U.S. government securities, and that U.S. government securities are debt securities, then it is reasonable to assume that the user believes that treasury bonds are debt securities. Using a KL-ONE type of knowledge representation for concepts, the conclusions of this rule are implicit: if there is a path of directed links from C to A, then A subsumes C.

Inheritance Rule

The inheritance rule also involves inferences about concepts the user believes.

Rule 12 *If the user believes a concept A has property P, and further believes that A subsumes concept B, then the user believes B has property P.*

For example, if the user believes that stock shares have a par value, and that a share of IBM is a share of stock, then it is reasonable to expect that the user believes a share of IBM has a par value.

Transitive Reasoning Rule

The transitive reasoning rule is similar to the transitive subsumption rule, but works with the rule dependency graph.

Rule 13 *If the user believes P_1 depends on P_2, and that P_2 depends on P_3, then the user believes that P_1 depends on P_3.*

This rule merely assumes that the user is able to chain together the individual dependencies he believes.

General Entailments

The entailment rules stated here are only explicit cases of more general entailments the user might perform. With a large body of existing knowledge, it is reasonable that the user will apply his logical reasoning capabilities to develop new beliefs. Unfortunately, the reasoning that humans perform is significantly different from formal logic. Thus it is not possible, for example, to apply an automatic theorem prover to a user model in order to discover further beliefs of the user. Such an approach would result in asserting to the user model many beliefs that the user did not have. Although it is quite likely that other entailment rules exist, they are not apparent. Many researchers are studying formal methods to model the knowledge and beliefs of agents [Fagin 85,Halpern 85,Konolige 83,Moore 84]. Progress in this area should lead to further, more robust entailment rules.

5.5 Human Behavior Rules

The human behavior rules are rules inspired by certain common aspects of human behavior. Many of these aspects are a result of the type of interaction that occurs between an advice-seeker and an expert, yet they are not general communicative rules. Instead, these rules are based on aspects of human behavior that go beyond maintaining a cooperative conversation. Five specific human behavior rules are presented in this section: the agent rule, the direct meta-statement rule, the evaluation rule, the shared reasoning rule and the misconception rule.

5.5.1 Agent Rule

The agent rule relates actions to beliefs held by the user. The basic premise of the agent rule is that an agent who performs an action knows the action, and the facts relating to the action. The fact that the user is an active agent is crucial to this rule. Knowing that the user knows of an action is not sufficient to conclude that he actually knows each step of the action, or the concepts related to the action. However, if the user actually performs the action, this is evidence that the user performed the steps of the action, and hence must know all about the action. The statement of the agent rule is:

Rule 14 *If the user is the agent of an action, then the user modelling module can attribute to the user knowledge about the action, the substeps of the action and the factual information related to the action.*

A situation in which the agent rule can be used is illustrated in the following statement:

(1) C. I just rolled over two CD's.

This statement will cause the user model to include the fact that the user believes she has performed the "roll over" action. Since the agent of this action is the user, the agent rule will cause the user model to include beliefs that the user knows about things related to rolling over a CD. These include: knowing that CD's come due, that money invested in one CD can be reinvested in another CD, that CD's are obtained from banks, and so on. Thus knowing the user was involved in an action can provide a lot of information about the user's beliefs.

5.5.2 Direct Meta-Statement Rule

Sometimes people will make direct statements about what they do or do not know. This is especially the case in the type of conversations studied in the transcripts, in which the advice-seeker and expert have never spoken before, and the caller lacks knowledge about the domain. Examples of a caller making a direct meta-statement about her knowledge include the following.

(2) C. I'm not too acquainted with finances,

(3) and I would like to have your advice.

Another technique for indicating a lack of knowledge is to simply ask a question. The following conversation includes two questions from the caller that indicate a lack of knowledge that he wants the expert to fill. The first question arises as a result of the conversation, while the second is initiated solely by the caller.

(4) E. OK. First five thousand should go into a money market fund,

(5) put a thousand in passbook ...

(6) C. OK. Can you tell me what the money market is?

(7) How does ...

(8) E. A money market fund is a group of people getting together—

(9) put their money together in a pool and it is invested by professional investors

⋮

(10) C. Can you explain to me what the certificate is?

(11) E. A certificate?

(12) C. Yes.

(13) E. I haven't said anything about a certificate.

(14) C. I know, I just ...

(15) E. All right. A certificate in a bank is evidence

(16) that you have from the bank that your money is on deposit for a specified period

⋮

The most appropriate way to handle direct statements by users about their own knowledge is to take them at their word. Although this may seem obvious, the problem is more difficult than a first glance might indicate. Users who claim a lack of knowledge about a subject frequently do have some knowledge about it. Often the statement about their lack of knowledge is a means of ensuring that the expert will not assume they know more than they do. Thus in subsequent interaction the expert may tell the user about things the user already knew. A user model that assumes the user does not know what he claims not to know will thus tend to be incorrect in parts. Some of these errors may be corrected by information about the user from other sources, but other information may still remain incorrect. However, since the user wants the system to believe he or she lacks knowledge about the subject, assuming that lack of knowledge in the user model should support acceptable interaction by the system.

The examples and discussion so far have ignored direct statements by the user about knowledge the user does have. While this should not be ruled out, such statements tend to be interpreted as a boast, and hence are rare in normal conversation.

With these preliminaries, the direct meta-statement rule can be stated as follows:

Rule 15 *If the user makes a direct statement about his or her own knowledge, the user modelling module should update the user model to reflect this statement.*

In terms of the concept taxonomy of knowledge in the user model, a statement such as "I don't know anything about C" should cause the direct meta-statement rule to indicate that the user does not know about C, the roles of C, any concepts subsumed by C, or elements of the rule dependency graph that are associated to C or the concepts it subsumes.

5.5.3 Evaluation Rule

The evaluation rule is concerned with reasoning the user has performed. Sometimes a user will make statements that indicate ignorance of the reasoning in the domain model. The system, in evaluating the information provided by the user, recognizes that actions taken by the user are incorrect. These incorrect actions indicate that the user did not know about certain rules in the domain knowledge base, for otherwise the user would not have made the incorrect actions. The formal rule is as follows:

Rule 16 *If the system is able to evaluate actions taken by the user given a certain situation, and those actions do not conform to the actions the system would have taken, then the user modelling module can identify portions of the reasoning done by the system that the user does not know about.*[7]

An example of how the evaluation rule is used can be seen in the following conversation.

(17) C. I have $10,000 in stocks and $10,000 in a savings fund.

(18) E. Why so much in a savings account?

[7]This rule assumes the system's knowledge is always correct and complete. Thus the user will never possess valid knowledge about the domain that the system does not know. In the context of advisory systems this assumption is reasonable, since it is expected that the expert is much more knowledgeable than the advice-seeker.

(19) C. Well, because I'm of uncertain health and I'm afraid I might be ill and need it.

(20) E. Well how about if you move it to a money market fund?

⋮

(21) E. There's no reason for leaving that much in a savings account.

(22) They must roll out the red carpet and kiss you on both cheeks every time you walk in there.

(23) Leave a maximum of $1000 in there and put $9000 into a money market fund.

In this conversation the caller makes a statement that leads the expert to an immediate evaluation: she has too much money in her savings account.[8] Thus in modelling the caller the expert can conclude that she does not recognize the reasoning that implies a lot of money in a savings account is not a good idea.

The evaluation rule only makes conclusions about what the user does not know. Thus, in the example, the rule would assert that the user does not know the rules in the domain model pertaining to investing in savings accounts. To find out what reasoning the user did use requires additional information. In the example, the expert pursues this reasoning by asking the caller to justify her action.

This rule does not work in situations where the user's action coincides with the expert's evaluation. In such situations it might seem that the user must know the reasoning rules the expert has, but this is not necessarily the case. The fact that the user's action coincides with the expert's evaluation could be accidental. Even if the user did perform reasoning in deciding on the action taken, that reasoning may not coincide with the expert's. Thus this situation does not warrant a judgment about the reasoning knowledge of the user.

5.5.4 Shared Reasoning Rule

The shared reasoning rule is similar to the evaluation rule. In this case though, the user explicitly states that he believes the expert and user share common reasoning. The following are a couple of examples of statements by an advice-seeker that assume some shared reasoning on the part of the advice-seeker and the expert.

(24) C. ...and here's where you're going to holler,

(25) I've got $5,000 in checking and $10,000 in savings.

The caller recognizes that the expert believes $5,000 in checking and $10,000 in savings is too much (probably from listening to the radio program earlier), and makes a statement that indicates he knows those amounts are too large.

In the second example, the caller simply says:

(26) C. I've got 350 shares of GM stock, need I say more?

[8]These conversations were transcribed in February 1982, when inflation was near its peak. At that time the interest rate on a savings account was considerably less than what could be earned in a money market fund, so having a lot of money in a savings account was always a bad idea.

At the time of the radio program the price for GM stock had fallen drastically. The caller is indicating that he believes this was a poor investment, and that the expert shares this belief.

How much reasoning can the user modelling module assume is shared? In the example above, it is not likely that the caller shares all of the complex reasoning rules the expert has for evaluating stocks. Rather, the caller knows some basic rules and the overall evaluation. This is incorporated into the shared-reasoning rule as follows:

Rule 17 *If the user makes a statement that assumes the user and system share reasoning about a subject, then the user modelling module can assert that the user believes that the system holds this reasoning, and that the overall reasoning of the user about this subject is the same as the overall domain reasoning rules.*

"Overall reasoning" should be interpreted to mean that the user model will not assume the user knows all of the intermediate decision properties and dependencies in the domain knowledge base. Rather, the properties believed by the user model to be known by the user are connected by positive or negative support links to the common conclusion. Thus the shared-reasoning rule allows the user model to have knowledge not only of the dependencies in the rule graph, but also of whether these links provide positive or negative support.

5.5.5 Misconception Rule

The misconception rule is unique. It is triggered by identifying that the user has a misconception about some information that the system knows. What distinguishes this rule is the fact that recognizing a misconception does not enable the user modelling module to draw new conclusions about the user. Thus, strictly speaking, the misconception rule is not an acquisition rule. It is included here because it enables certain activities that can result in the acquisition of further information for the user model.

Misconceptions play an important role in advisory interactions. If the expert recognizes a misconception on the part of the advice-seeker, the expert will usually seek to correct this misconception immediately, before continuing with the conversation. Thus a user modelling module capable of recognizing such misconceptions is an important component of a cooperative advisory system.

In addition to aiding the advisor, misconception recognition can eventually lead to additional information for the user model. Because the advisory application will try to correct a misconception held by the user, notifying the application of a user misconception can lead to a clarification dialogue that will resolve the misconception in the user model. Thus the misconception rule is part of a two stage process. The rule will detect the misconception and notify the application. From this point the responsibility lies with the application to resolve the misconception. The application may be able to do this directly, using its own knowledge and reasoning ability. In this case the user modelling module should provide facilities to allow the application to update the user model (or at least present the corrective information). If the application cannot resolve the misconception on its own, it may then initiate a dialogue with the user that will allow the user model to be updated via the normal acquisition rules.

An example of a dialogue in which the advice-seeker holds a misconception that the expert seeks to correct is the following:

(27) C. How about if I bought it from a major bank?

(28) Can I give the name of the bank I'm using (on the radio) or not?

(29) E. Yes, go ahead.

(30) C. I'm using Bache and Company.

(31) E. They're not a bank—they're a broker.

In this dialogue the caller has a rather clear misconception, which the expert immediately corrects.

The misconception rule says:

Rule 18 *If the user model indicates the user believes some piece of information P is true, while the domain model knows P is false, or vice versa, then the user modelling module can notify the application that the user holds a misconception about P.*

Note that a misconception is detected only when the user model indicates the user has a definite belief about P. Lack of knowledge by the user is not sufficient to indicate a misconception.

For the type of knowledge being considered in this work, four kinds of misconceptions can occur. Two, misclassification and misattribution, are object-related misconceptions. The other two, spurious dependencies and incorrect support, are related to reasoning rules.

Object-Related Misconceptions Object-related misconceptions are misconceptions about the knowledge in the domain concept hierarchy.[9] A *misclassification* occurs when the user model indicates that a subsumption relationship exists between two concepts, but that relation does not occur in the domain model. In a typical example, the user might believe that a whale was a kind of fish. A *misattribution* occurs when the user model ascribes a role to a concept that the domain model indicates the concept does not have. Thus if the user believes that a whale has gills, the property of having gills is incorrectly attributed to the concept of whale.

Reasoning-Related Misconceptions The misconceptions related to reasoning rules concern the dependency links between properties in the rule graph. A *spurious dependency* occurs when the user believes that a decision property A influences another decision property B, and the domain model knows that B does not depend on A, or vice versa. Such misconceptions can usually be detected when the user's evaluation of a situation differs from that of the expert. The *incorrect support* misconception occurs when the user model and domain model both agree about a dependency link between two concepts, but conflict about whether the link provides a positive or negative support.

5.6 Statements Made by the System

The user model acquisition rules in the preceeding sections have used the user's statements and the existing user and domain models to draw further conclusions about the user. In this section, information that can be obtained for the user model from the statements made by the system is focused on. For the most part, the contributions of the system do not add much to the user model. Consequently new rules will not be introduced. Instead, some of the acquisition rules already presented will be modified to apply to system statements as well as user statements.

Two assumptions are appropriate in considering how system statements should influence the user model. The first assumption is that the user is paying attention to the statements made by the system, since the user came to the system for advice. The second assumption is that the user

[9]The notion of object-related misconceptions was introduced by McCoy. The descriptions given here for misclassification and misattribution are taken from [McCoy 85].

believes what the system says. This is reasonable for two reasons. First, the system itself should be generating statements that the user can be expected to understand and believe. Secondly, since the user is seeking advice of an expert, the user is likely to believe what the expert says. Even when this assumption fails, the user is likely to challenge statements he does not believe, perhaps by asking follow-up questions. In this case, the user model can be corrected.

The inferences that can be drawn from statements made by the system are positive inferences: the user model will receive new information about what the user does believe, but not information concerning lack of knowledge on the part of the user. These positive inferences are triggered by information in the system statement, just as the communication rules described earlier are triggered. In fact, several of the communication rules can be altered slightly to recognize that the source of information can be either a user or a system statement.

5.6.1 Revised Direct Inference Rules

Each of the direct inference rules can be applied to system statements as well as user statements. Recall that the direct statement and presupposition rules rely only on the statement itself, and do not depend upon the user or domain models. The direct statement rule says the user believes a statement. If the user makes the statement this is definitely reasonable. If the system makes a statement, the rule is also reasonable, assuming the user believes what the system says. The presupposition rule states that the user believes the presuppositions of a statement. For the user to understand and believe a statement made by the system, he must also believe the presuppositions of that statement. Thus the direct inference rules can be modified to apply to statements made by the system as well. The revised rules are:

Rule 1′ *The user modelling module can assume that a statement made by* either the user or the system *is believed by the user.*

Rule 2′ *The user modelling module can assume that presuppositions of statements made by* either the user or the system *are believed by the user.*

5.6.2 Revised Implicature Rule

Only one of the implicature rules can be applied to statements made by the system. This is the relevancy rule. The relevance rule states that the user believes all of the information provided in a statement is relevant. If the user believes the system is an expert, this rule should apply not only to statements made by the user, but to also the system's statements. Thus the relevancy rule can be restated as follows:

Rule 3′ *If a statement includes the clause P, the user modelling module can assume that the user believes that P, in its entirety, is used in reasoning about the current goal or goals of the interaction.*

The remaining communication rules cannot be applied to system statements because they do not make positive inferences about the user's beliefs. The model-based and human behavior rules are based on sources of knowledge other than the communication between system and user, so they cannot be extended to apply to statements made by the system either. Thus knowledge for the user model obtainable from system statements can be handled by simply extending some of the communication rules previously presented.

5.6.3 Embedding Acquisition Rules in the User Model

The rules that have been presented in this chapter have focused on acquiring what might be call *primary knowledge* for the user model: the user's own beliefs about the domain. Another important area of user model acquisition is acquiring knowledge about what the user believes about the knowledge of other agents, in particular the system. Although acquiring this knowledge will require some specialized techniques, user's beliefs about other agents can, in part, be acquired by a re-application of the rules discussed in this chapter.

The user's model of other agents can be built by recursively applying the model acquisition rules. For example, consider building the model the user has of the system. If the user uses the acquisition techniques presented in this chapter, these rules can be applied to statements made by the system in order to construct the user's model of the system. In fact, this process can be continued to arbitrarily deep levels of modelling.

A more robust modelling system would allow for the acquisition rules used by these embedded models to differ from the system acquisition rules. This is particularly important if domain specific model acquisition rules are included. In this case it is quite likely that the acquisition rules of various agents would differ, requiring the rules to be contained in the user model itself.

Chapter 6

Proposal for Research

The preceeding chapters have made a case for a user modelling module with some features significantly different from those in existing user modelling systems. I have argued that general user modelling systems are needed, and have focused on achieving domain generality in a user modelling module. The effectiveness of a general user modelling module depends on its ability to do implicit user model acquisition. Domain generality requires that a user modelling module have the ability to acquire its own information. This allows the module to easily transfer between systems that have different underlying domains. Implicit user model acquisition also reduces (or even eliminates) the effort required to explicitly encode information about users for the user model ahead of time. In the previous chapter I proposed a set of rules that should enable implicit user model acquisition, yet are domain independent.

This chapter describes my future research plans. This work will be undertaken with two goals in mind. The first (and by far the more important) goal is to test the effectiveness of the implicit user model acquisition rules. The second goal is to implement a general user modelling module for cooperative advisory systems. I hope to present a framework for the general user modelling module, but will not be able to implement all the pieces. These goals will be discussed in the remainder of this chapter, in reverse order. The next section includes the framework for the general user modelling module, my current ideas for its implementation, and a discussion of which portions will actually be implemented. The following section will discuss implementing and testing the user model acquisition rules. The final section reviews the contributions this work will make to the field of user modelling.

6.1 General User Modelling Module Architecture

A general user modelling module is part of a system that has at least three components:

1. An application. This is that portion of the system that accomplishes the task for which the system was built. In the systems we have been considering this is the expert system advisor.
2. A user interface. The user interface manages the communication between application and user. Presumably the user interface will perform translations so that the user's statements are understandable by the application, and vice versa.
3. The user modelling module. This builds the model of the user by eavesdropping on the interaction between the user interface and the application, and by using the domain knowledge of the application.

The user modelling module itself has three components, which can be describe as the representation facilities, acquisition facilities, and interface support facilities. Each of these will be discussed in the following sections, followed by a discussion of how explicit and implicit knowledge acquisition techniques could be coordinated in a general user modelling module.

6.1.1 Representation Facilities

The user modelling module must be able to represent the information believed to be held by the user. The representation requires some form of knowledge representation system or language. In addition, as discussed in section 2.3.4, the representation for a general user modelling module must be able to handle recursive belief structures (to represent the beliefs a user might hold about other agents, including the system) and must be able to deal with the inherent non-monotonicity of the information in the user model.

As a basis for representing the user's knowledge of factual information, I plan to use NIKL (New Implementation of KL-ONE) [Moser 83]. NIKL uses a semantic network representation with a restricted set of link types. I will be primarily interested in representing concepts and their roles, along with the subsumption links between concepts. The classifier will not be used.

One problem with building a user modelling module that must maintain models for a large number of individual users is the large amount of storage space required. Furthermore, much of the information in each of the user models is effectively the same. Kobsa [Kobsa 85] has presented a representation technique that solves this problem, and at the same time effectively deals with the problem of representing recursive belief structures as well.

Kobsa represents the information for all users in one knowledge base. This information is partitioned to indicate the beliefs of individual users through the use of *contexts.* A context is a set of *acceptance attitudes.* An acceptance attitude indicates the state of belief an individual user model holds for an element of the knowledge base. In Kobsa's VIE-DPM system the acceptance attitude can have three values:

'+' The user believes (accepts) the item.

'–' The user believes the item is not true.

'0' The user has no belief about the item.

The elements of the knowledge base to which the acceptance attitudes refer consist of each of the isolated components in the knowledge representation structure. Thus there will be acceptance attitudes for each concept, each role, each role filler, each of the subsumption links in the hierarchy, and so on. Every context holds one acceptance attitude for each of the elements in the central knowledge base. A context thus represents a user's attitudes towards all of the elements in the central knowledge base.

The context technique allows not only the individual user models to share information, but the domain model as well. The representation of the domain information is simply another context, like all the rest. Thus the context technique provides a convenient mechanism for representing perturbation user models. The underlying collection of knowledge in the knowledge base may be somewhat jumbled and even contradictory, but that does not matter. The "correct" knowledge (from the application's point of view) is obtained through the domain model context. Other contexts are free to be as similar to or different from the domain model as they like.

Kobsa also uses the acceptance attitude mechanism to handle beliefs the user may have about other individual's beliefs. This is accomplished by including in the context acceptance attitudes that refer to the acceptance attitudes of other contexts. Thus if *A* believes that whales are fish,

A's context will include an acceptance attitude with a '+' indication, referring to the subsumption link between fish and whales. If B does not believe that A believes whales are fish, B will have an acceptance attitude with a '–' link referring to the above mentioned acceptance attitude of A.

In implementing the general user modelling module I hope to adopt a technique similar to Kobsa's. My implementation will differ from Kobsa's in two ways. First, it must be extended to handle the representation of rule-based knowledge. Fortunately this is a simple extension. Secondly, my implementation will take a different approach to handling the acceptance attitudes.

Rule-Base Knowledge

The rule-base I envision for the domain model will be a simple one. Rules will be limited to a single antecedent and consequent per rule. This limitation is necessary for the user modelling module to be able to identify the elementary beliefs of the user. More complex rules can lead to confusion: the user may not believe the rule, but may believe significant portions of it, or vice versa. Although this limitation is strict, it is in keeping with other observations that have been made about rule-based systems. A common criticism of conventional rule-based systems is that they mix domain and control knowledge in a single rule. This mixing severely limits the ability of the system to adequately explain its reasoning, or to be used in teaching [Clancey 83,Brown 80,Swartout 83].

Given the limitation to simple rules, extending acceptance attitudes to the rule base is a simple matter. The rule base can be organized into a dependency graph with properties as the nodes and rules forming the links. The links will be marked to indicate the form of the dependency. For the sake of user modelling, a simple indication of positive or negative support seems sufficient. However, the support indicator range could be greater, such as evidence weights. A context will include acceptance attitudes for each node, each rule link and for the support indicator on each link.

Acceptance Attitudes

Kobsa limits acceptance attitudes to a three-valued logic. Rather than simply indicate belief or disbelief in an item, I plan to keep a record of the justifications for holding a belief as well. This extension of the acceptance attitudes addresses the last issue in representing user models, the issue of non-monotonicity in the model.

Keeping justifications for the beliefs held in a user model allows those beliefs to be altered if necessary. Such modifications are generally handled by a *truth maintenance system* [Doyle 79]. A truth maintenance system seeks to maintain consistent sets of beliefs, and at the same time cache inferences that are performed to reduce computation. The justifications for belief in an item, p, will consist of the acquisition rules that affect the user model's acceptance of the fact. This can include justifications both for and against the user's belief in p.

6.1.2 Acquisition Facilities

The acquisition facility consists of the user model acquisition rules, and the machinery necessary to support them in the user modelling module. Three features of the acquisition facilities are significant: the independence of the acquisition rules, the need for rule arbitration, and the location of the rules in the user modelling module.

Rule Independence

Although the execution of some rules may enable other rules to fire, the rules as described in the previous chapter are strictly independent of each other. This independence means that the rules may be run in any order, or even in parallel. Each rule only runs when it recognizes a configuration of knowledge in its knowledge sources (the user's statement, the system's statement, the existing user model and the domain model) that it is capable of handling.

In practice the acquisition will be driven by the arrival of new information from the system-user interaction. A new statement, either from the user or the system will tend to activate communication rules first. Once some of these have fired, changing the user model, the model-based rules and human behavior rules are more likely to fire. This process will continue until no further rules can fire.

Ideally, the control of acquisition rules should be both data and demand driven. Because a user modelling module may need to volunteer information to the application, some model acquisition must occur as new data arrives. On the other hand, this may result in a large amount of wasted computation, particularly when the user modelling module draws inferences that are subsequently reversed when as information arrives. However, since this system is primarily concerned with testing the effectiveness of the acquisition rules, I plan to adopt the simpler, data-driven technique.

Rule Arbitration

Despite the independence of the rules, there must be some technique for reasoning about the precedence order of the rules. This arbitration is necessary when trying to determine acceptance attitudes in individual user models. It is quite possible that different acquisition rules may conflict about whether the user believes a particular item or not. For example, the evaluation rule may support the opinion that the user does not know a decision property p, while the consequential generalization rule might suggest that the user believes p. If it is necessary to know whether the user believes p, some means of reasoning about the acquisition rules is necessary.

The user modelling module may need to determine the acceptance attitude towards a particular item for two reasons. An obvious situation is when the application (or some other module outside the user modelling module) asks to know whether the user believes some item p. In this case the user modelling module must make a determination about the user's belief of p in order to meet the request.[1] The model acquisition process itself may also need to know acceptance attitudes. Many of the model acquisition rules, particularly the model-based rules, are triggered by certain collections of beliefs in the user model. For these rules to fire, assumptions must be made about the belief status of items in the user model.

At present I have given little thought to the actual arbitration techniques that will be used. It seems that a certain precedence among the rules exist. For example, conclusions based on the communication rules seem more likely to be correct because they are based on objective evidence of the user's beliefs—the user's behavior. On the other hand, in some situations the arbitration may be dependent on particular circumstances that require further reasoning. To accommodate this possibility I anticipate that a set of arbitration rules will also be included in the user modelling module. These rules can easily handle simple precedence relations among the acquisition rules, while enabling more complex arbitration reasoning if necessary.

[1]This is not totally correct. The user modelling module might simply sidestep the issue by responding to the request by providing a list of the justifications concerning the user's belief in p.

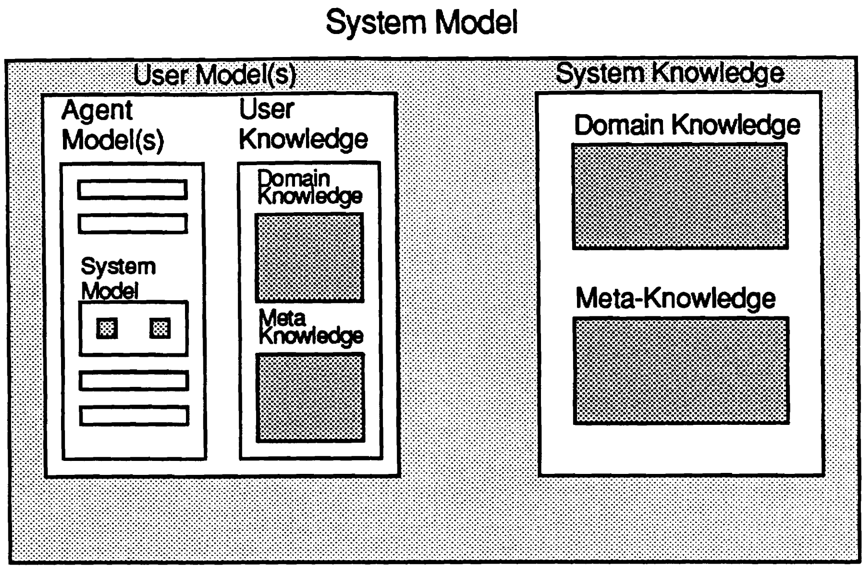

Figure 6.1: The location of knowledge sources in system and user models

Rule Location

The location of the acquisition (and arbitration) rules in the overall system is also an important issue. The knowledge a user uses to reason about what he knows is an important component of a user model, as pointed out in chapter 2. This meta-knowledge may vary among users, thus each user model may have unique acquisition and arbitration rules. The presence of meta-knowledge in the user model suggests a parallel between the structure of the user and system models. The system's knowledge consists of knowledge about its users, the domain, and meta-knowledge for reasoning about both. Likewise a user model contains the user's knowledge about the domain, other agents (including the system), and meta-knowledge. This relationship is illustrated in figure 6.1. The parallel between user and system models can be exploited, so that both share the same representation and reasoning facilities.

The user's meta-knowledge will not be emphasized here. Acquiring knowledge about the user's meta-knowledge is very difficult, since direct evidence is usually not available. In this work, I will assume that the user's meta-knowledge about acquiring models of other agents is the same as the system's.

6.1.3 Interface Support Facilities

The third component of a general user modelling module is the interface support facilities. These facilities control the interaction between the user modelling module and other modules of the system. If the user modelling module is to be truly general, it must also be truly modular. This means that a well-defined interface must exist for communication between the user modelling module and other modules of the system, and that all modules are limited to using this interface.

The user modelling module interface must support a variety of types of communication. The user modelling component must be able to access the communications between the system and the user, as well as the domain knowledge of the system. On the other hand, modules in the system may need to request information from the user model, or directly assert information about the user to the knowledge base. Furthermore, the user modelling module may need to volunteer information to make provisions in the general user modelling module for a stereotype hierarchy.

6.2 Testing the Acquisition Rules

The primary goal of my future research is to test the effectiveness of the implicit user model acquisition approach, and of the acquisition rules outlined in this paper. This testing will involve the implementation of the acquisition portion of a general user model, the building of a sample domain model to use as a basis for the testing, and the examination of the user model during and after the processing of simulated conversations between the system and user. In this section I shall describe these three stages.

6.2.1 Implementing the Acquisition Rules

The acquisition rules as presented in the preceeding chapter are still in a rough form. A significant portion of the work ahead will be devoted to refining these acquisition rules to the point where they can be implemented in a software system. In this process the formulation and organization of the rules may change. In particular, it is likely that many of the rules presented here will be implemented as a group of rules in the actual system. Likewise, it is possible that some rules may be combined or eliminated during the implementation process.

Once the acquisition rules have been refined, the arbitration rules must be implemented as well. This process may require more experimentation, since the interaction of various rules can become quite complex and difficult to predict. At this point it will also be possible to see whether any broad classifications for the arbitration rules can be discovered, classifications that would be useful in defining new arbitration rules when new acquisition rules are added.

In the previous section the architecture and control structure for the acquisition rules were described. This control structure, together with the refined acquisition and arbitration rules will be sufficient to implement the model acquisition portion of a general user model.

6.2.2 The Domain

An underlying application domain will need to be built to support the testing of the acquisition rules. The domain planned is that of personal investments. This domain is attractive for several reasons. First, personal investing was the domain of discussion in the radio program transcripts studied in developing the acquisition rules, facilitating the creation of sample conversations that can be used to test the performance of the acquisition rules. This domain will also enable a comparison of the modelling capabilities of the system with those of the human expert in the transcripts.

Second, people vary greatly in their knowledge about investments. Thus modelling the range of possible users of an investment advisory system will be a good test of the capabilities of the user modelling module.

Finally, personal investing is also a reasonable domain for a cooperative advisory system. Several software systems, including expert systems, exist that are designed to aid investors in choosing securities to invest in. Interfaces capable of dealing with a variety of individuals with different levels of knowledge about the domain will be needed in such a system, and so will user modelling capabilities.

6.2.3 Simulating Conversations

The actual effectiveness of the user model acquisition rules will be tested by letting the user modelling module process several simulated conversations. These conversations will be created by to other modules in the system, or to notify other modules when information changes.

I will focus little attention on the interface support issues. Although important, many of the interface support issues are far removed from the problems of user model acquisition. Thus even simple capabilities, like support for external query of the user model, will not receive the attention they deserve.

In one respect my implementation will depart from being a truly modular system component. The advantages of sharing the knowledge representation for both the domain and user models is too great to ignore. Thus domain and user models will not be strictly separate entities. Instead, the domain model is constrained to use the same knowledge representation language and utilities that the user modelling module does. This departure from strict modularity seems acceptable in light of the advantages obtained in sharing the knowledge base.

6.1.4 Explicit and Implicit Acquisition

The emphasis of my research is on implicit user model acquisition. A practical user modelling system, however, will probably need to use both explicit and implicit techniques to acquire knowledge about the user. This section will briefly discuss how these two activities could be coordinated to achieve more powerful user modelling capabilities.

As mentioned in section 3.1.2, the use of stereotype modelling in a user modelling system can be very effective in identifying a large set of beliefs about the user quickly. Stereotypes represent the compiled knowledge that comes from experience. If this knowledge can be integrated with the implicit acquisition techniques, without introducing the need for pre-encoding large amounts of knowledge in the user model, the performance of the user modelling module will be improved.

I propose is to use stereotypes as a supplement to the implicit model acquisition techniques. Stereotypes will be treated as a "last resort" when attempting to determine beliefs of the user. This approach is justifiable for the following reason. Most of the implicit acquisition rules are conservative. They are more likely to miss beliefs held by the user than to attribute beliefs to the user falsely. Also, several of the rules (such as the consequence rule) conclude that the user believes one or more of a small set of items, without making further claims about which particular item or items the user believes. Thus in many cases the implicit model acquisition rules may get close to answering a request for information, but not be able to make a final determination from among a set of possibilities. At this point the compiled knowledge of a stereotype can be helpful, indicating which of the possibilities is most likely to be true for this individual. Reasoning about user beliefs is thus a two-stage process. In the first stage the implicit acquisition rules are used in an attempt to find a solution. If this fails, stereotypes can be used to pick up where the implicit rules left off.

Implicit user model acquisition and the use of stereotypes can be integrated in another way as well. Stereotypes do not necessarily have to be acquired explicitly. The powerful implicit acquisition rules can be used to help build stereotype models specific to the system as it is used. If the system has a large body of individual user models, these can be abstracted to create general stereotypes for classes of users. Such a technique would not only improve performance, but increase storage efficiency as well. A hierarchy of stereotypes as discussed in [Finin 86] could be used to organize both the individual and stereotype user models. The user model acquisition process would then consist of two steps: applying the implicit acquisition rules, followed by possible classification (or re-classification) to identify the most appropriate stereotype for the user. In fact, these steps might be iterated, since selecting a new stereotype might initiate additional domain-based rules, as well as acquisition rules specific to that stereotype.

The coordination of the implicit acquisition rules is an interesting and potentially fruitful area. Unfortunately this work will not have a chance to address the issue. The most that can be done is hand-coding the MRL representations of statements by the advice-seeker and expert. Since the planned domain is that of personal investments, these conversations will be based on the transcripts of the Harry Gross radio program.

The effectiveness of the acquisition rules will be judged in two ways. The first method is to study the user model created by the system as the result of processing a conversation. Since the transcripts provide a large number of examples of expert responses to advice-seekers' statements, the model that a human expert develops of the advice-seeker can be inferred in some detail. Furthermore, humans who read the conversations to be processed by the user modelling module will have definite expectations for the information the expert should believe about the advice-seeker. These expectations can be compared to the actual results obtained by the user modelling module. The performance of the user model acquisition rules can thus be measured by comparing the model obtained by applying these rules to the model humans would expect an expert to develop.

The second method for judging the effectiveness of the acquisition rules is to study the impact of the user model they generate on the behavior of the overall system. If the rules are effective, the system behavior produced by using the user model should be better than the behavior when a user model is not used. The determination of what constitutes better behavior is largely subjective, but in many situations generally acceptable criteria can be identified. For example, rather than generate explanations of its reasoning solely from the domain knowledge base (as conventional expert systems do), the user model might enable the system to generate explanations in terms of concepts the user model indicates the user understands. Likewise, the system might use knowledge from the user model in deciding what questions to ask the user: if the user model indicates the user does not know about certain properties, the system might omit asking questions about them. Thus if the user model acquisition rules acquire information about the user that enables the overall system to improve its interaction with the user, this will be evidence that the acquisition rules are useful in an interactive system.

I have also claimed that these user model acquisition rules are domain independent. If time permits I would like to test this hypothesis as well, by changing the underlying domain and letting the user modelling module process a conversation that is concerned with a different domain. I have no particular domain in mind for this second test, but expect that it would be a much simpler or less complete domain than that implemented for personal investments.

6.3 Contributions

In summary, the work presented here will make three contributions to research in user modelling. The first contribution is the demonstration that powerful implicit user model acquisition techniques are possible. This means that a user model may be implemented more easily in many systems, because the amount of hand-coded user model information can be reduced.

The second contribution of this research is that implicit user model acquisition can be performed in a domain independent manner. Thus implicit user modelling techniques not only reduce the amount of explicit knowledge about users that must be encoded, they can be readily transferred to similar systems that operate in a different domain. This means that a general user modelling module is feasible. Such a module can be included in an overall system with a minimum of disruption to enhance the interaction capabilities of the system.

The final contribution of this research is the presentation of an architecture for a general user modelling module. Although it will not be possible to implement all portions of this module, implementation that does occur will be done in the context of this architecture. Thus the general user modelling module may be expanded with time to encompass its complete capabilities.

Appendix A

Summary of Implicit Acquisition Rules

Direct Statement Rule The user modelling module can assume that a statement made by either the user or the system is believed by the user.[1]

Presupposition Rule The user modelling module can assume that presuppositions of statements made by either the user or the system are believed by the user.

Relevancy Rule If a statement includes the clause P, the user modelling module can assume that the user believes that P, in its entirety, is used in reasoning about the current goal or goals of the interaction.

Sufficiency Rule If the user omits a relevant piece of information from a statement, then either the user does not know of that piece of information, does not know whether that information is relevant to his current goal or goals, or does not know the value for the piece of information.

Ambiguity Rule If the user makes a statement that the system finds ambiguous in the current context, then the user lacks knowledge of one or more of the alternative meanings for his statement.

Action Rule If the user model includes the belief that a user knows an action, then the the user modelling module can attribute to the user knowledge of the preconditions and postconditions of that action.

Consequence Rule

i If the user believes a fact that can only result from one of a small set of actions, then the user believes that one or more of those actions has occurred.

ii If the system knows a fact that can only result from one of a small set of actions, then the system believes that one or more of those actions has occurred.

Concept Generalization Rule If the user model indicates that the user knows several concepts that are specializations of a common, more general concept in the domain model, the user modelling module may conclude that the user knows the more general concept, and the

[1]The direct statement, presupposition, and relevancy rules stated here reflect the modifications added to handle statements made by either the user or the system.

subsumption relationship between the more general concept and the more specialized concepts as well.

Consequential Generalization Rule If the user model indicates that the user believes that several properties that have a common consequence are relevant, then the user modelling module may assert that the user knows of the common consequent, believes that it is relevant and knows of the links between the set of properties and the common consequent.

Distribution Rule Given a situation in which the user is to provide a list of items from a class for a response, if the user's response consists only of a particular subset of those items, then the user modelling module can assert that the user lacks knowledge of the other items in the class.

Transitive Subsumption Rule If the user believes A subsumes B, and that B subsumes C, then the user believes that A subsumes C.

Inheritance Rule If the user believes a concept A has property P, and further believes that A subsumes concept B, then the user believes B has property P.

Transitive Reasoning Rule If the user believes P_1 depends on P_2, and that P_2 depends on P_3, then the user believes that P_1 depends on P_3.

Agent Rule If the user is the agent of an action, then the user modelling module can attribute to the user knowledge about the action, the substeps of the action and the factual information related to the action.

Direct Meta-Statement Rule If the user makes a direct statement about his or her own knowledge, the user modelling module should update the user model to reflect this statement.

Evaluation Rule If the system is able to evaluate actions taken by the user given a certain situation, and those actions do not conform to the actions the system would have taken, then the user modelling module can identify portions of the reasoning done by the system that the user does not know about.

Shared-Reasoning Rule If the user makes a statement that assumes that the user and system share reasoning about a subject, then the user modelling module can assert that the user believes that the system holds this reasoning, and that the overall reasoning of the user about this subject is the same as the overall domain reasoning rules.

Misconception Rule If the user model indicates the user believes some piece of information P is true, while the domain model knows P is false, or vice versa, then the user modelling module can notify the application that the user holds a misconception about P.

Bibliography

[Allen 80] Allen, James F. and Perrault, C. Raymond. Analyzing Intention in Utterances. *Artificial Intelligence* 15:143–178, 1980.

[Allen 82] Allen, James F., Frisch, Alan M., and Litman, Diane J. ARGOT: the Rochester Dialogue System. In *Proceedings of the National Conference on Artificial Intelligence*, pages 66–70. 1982.

[Anderson 85] Anderson, John R., Boyd, C. Franklin, and Yost, Gregg. The Geometry Tutor. In 9th *International Conference on Artificial Intelligence*, pages 1–7. 1985.

[Brachman 85a] Brachman, R. J., Pigman, V. G., and Levesque, H. J. An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON. In 9th *International Conference on Artificial Intelligence*, pages 532–539. William Kaufmann, Inc., Los Altos, California, 1985.

[Brachman 85b] Brachman, R. J. and Schmolze, J. G. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9:171–216, 1985.

[Brown 75] Brown, J. S., Burton, R., Miller, M., DeKleer, J., Purcell, S., Hauseman, C., and Bobrow, R. *Steps Towards a Theoretical Foundation for Complex, Knowledge-Based CAI.* Technical Report 3135, Bolt, Beranek and Newman, 1975.

[Brown 78] Brown, J. S. and Burton, R. R. Diagnostic Models for Procedural Bugs in Basic Mathematical Skills. *Cognitive Science* 2:155–192, 1978.

[Brown 80] Brown, John Seely and VanLehn, Kurt. Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science* 4:379–426, 1980.

[Burton 82a] Burton, Richard R. Diagnosing Bugs in a Simple Procedural Skill. In Sleeman, D. and Brown, J. S. (editors), *Intelligent Tutoring Systems*, pages 157–184. Academic Press, New York, 1982.

[Burton 82b] Burton, Richard R. and Brown, John Seely. An Investigation of Computer Coaching for Informal Learning Activities. In Sleeman, D. and Brown, J. S. (editors), *Intelligent Tutoring Systems*, pages 79–98. Academic Press, New York, 1982.

[Carberry 83] Carberry, Sandra. Tracking User Goals in an Information Seeking Environment. In *Proceedings of the National Conference on Artificial Intelligence*, pages 59–63. 1983.

[Carr 77] Carr, Brian and Goldstein, Ira P. *Overlays: A Theory of Modelling for Computer Aided Instruction.* Technical Report A. I. Memo 406, MIT Artificial Intelligence Laboratory, Cambridge, Massachusetts, 1977.

[Chin 86] Chin, David N. *KNOME: Modeling What the User Knows in UC.* Technical Report ?, Department of EECS, University of California, Berkeley, 1986.

[Clancey 82] Clancey, William J. Tutoring Rules for Guiding a Case Method Dialogue. In Sleeman, D. and Brown, J. S. (editors), *Intelligent Tutoring Systems*, pages 201–226. Academic Press, New York, 1982.

[Clancey 83] Clancey, W. J. The Epistemology of a Rule-Based Expert System – A Framework for Explanation. *Artificial Intelligence* 20:215–251, 1983.

[Davis 84] Davis, Randall and Buchanan, Bruce G. Meta-Level Knowledge. In Buchanan, Bruce G. and Shortliffe, Edward H. (editors), *Rule-Based Expert Systems.* Addison-Wesley, Reading, MA, 1984.

[Doyle 79] Doyle, Jon. A Truth Maintenance System. *Artificial Intelligence* 12(3):231–272, 1979.

[Fagin 85] Fagin, Ronald and Halpern, Joseph Y. Belief, Awareness and Limited Reasoning: Preliminary Report. In 9th *International Conference on Artificial Intelligence*, pages 491–501. 1985.

[Finin 86] Finin, Tim and Drager, David. GUMS$_1$: a General User Modelling System. In *Proceedings of the 1986 Conference of the Canadian Society for Computational Studies of Intelligence*, pages 24–30. 1986.

[Genesereth 79] Genesereth, Michael. The Role of Plans in Automated Consultation. In 6th *International Conference on Artificial Intelligence*, pages 311–319. 1979.

[Genesereth 82] Genesereth, Michael R. The Role of Plans in Intelligent Teaching Systems. In Sleeman, D. and Brown, J. S. (editors), *Intelligent Tutoring Systems*, pages 137–156. Academic Press, New York, 1982.

[Grice 75] Grice, H. P. Logic and Conversation. In Cole, P. and Morgan, J. L. (editors), *Syntax and Semantics.* Volume 3. Academic Press, New York, 1975.

[Grosz 86] Grosz, Barbara J. and Sidner, Candace L. Attentions, Intentions, and the Structure of Discourse. *Computational Linguistics* 12(3):175–204, 1986.

[Halpern 85] Halpern, Joseph Y. and Moses, Yoram. A Guide to the Modal Logics of Knowledge and Belief: Preliminary Draft. In 9th *International Conference on Artificial Intelligence*, pages 480–490. 1985.

[Hoeppner 83] Hoeppner, Wolfgang, Christaller, Thomas, Marburger, Heinz, Morik, Katharina, Nebel, Bernhard, O'Leary, Mike, and Wahlster, Wolfgang. Beyond Domain Independence: Experience with the Development of a German Language Access System to Highly Diverse Background Systems. In 8th *International Conference on Artificial Intelligence*, pages 588–594. 1983.

[Johnson 84] Johnson, W. Lewis and Soloway, Elliot. Intention-Based Diagnosis of Programming Errors. In *Proceedings of the National Conference on Artificial Intelligence*, pages 162–168. 1984.

[Johnson 85] Johnson, William Lewis. *Intention-Based Diagnosis of Errors in Novice Programs.* PhD thesis, Department of Computer Science, Yale University, 1985.

[Joshi 82] Joshi, Aravind K. Mutual Beliefs in Question Answering Systems. In Smith, N. (editor), *Mutual Belief.* Academic Press, New York, 1982.

[Joshi 84] Joshi, A., Webber, Bonnie, and Weischedel, Ralph. Preventing False Inferences. In *Proceedings of the Tenth International Conference on Computational Linguistics*, pages 134–138. 1984.

[Kaplan 82] Kaplan, S. J. Cooperative Responses from a Portable Natural Language Databas Query System. *Artificial Intelligence* 19(2):165–188, 1982.

[Kass 86] Kass, Robert and Finin, Tim. *The Role of User Models in Question Answering Systems.* Technical Report MS-CIS-86-63 (Linc Lab 30), Department of Computer and Information Science, University of Pennsylvania, 1986.

[Kass 87] Kass, Robert and Finin, Tim. Modelling the User in Natural Language Systems. *Computational Linguistics* Special issue on User Modelling:to appear, 1987.

[Kobsa 84] Kobsa, Alfred. Three Steps in Constructing Mutual Belief Models from User Assertions. In *Proceedings of the* 6th *European Conference on Artificial Intelligence*, pages 423–427. 1984.

[Kobsa 85] Kobsa, Alfred. Using Situation Descriptions and Russellian Attitudes for Representing Beliefs and Wants. In 9th *International Conference on Artificial Intelligence*, pages 513–515. 1985.

[Kobsa 86] Kobsa, Alfred. *A Taxonomy of Beliefs and Goals for User Models in Dialog Systems.* Technical Report ?, Department of Computer Science, University of Saarbrucken, 1986.

[Konolige 83] Konolige, Kurt. A Deductive Model of Belief. In 8th *International Conference on Artificial Intelligence*, pages 377–381. 1983.

[Lipkis 82] Lipkis, T. *A KL-ONE Classifier.* Technical Report 4842, Bolt, Beranek and Newman, 1982.

[London 82] London, Bob and Clancey, William J. Plan Recognition Strategies in Student Modelling: Prediction and Description. In *Proceedings of the National Conference on Artificial Intelligence*, pages 335–338. 1982.

[McCoy 85] McCoy, Kathleen Filliben. *Correcting Object-Related Misconceptions.* Technical Report MS-CIS-85-57, Department of Computer and Information Science, University of Pennsylvania, 1985.

[McKeown 85] McKeown, Kathleen R. Tailoring Explanations for the User. In 9th *International Conference on Artificial Intelligence*, pages 794–798. 1985.

[Moore 84] Moore, Robert C. A Formal Theory of Knowledge and Action. In Moore, R. C. and Hobbs, J. (editors), *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing, Norwood, NJ, 1984.

[Moser 83] Moser, M. G. *An Overview of NIKL, The New Implementation of KL-ONE.* Technical Report 5421, Bolt, Beranek and Newman, 1983.

[Murray 85] Murray, William R. Heuristic and Formal Methods in Automatic Program Debugging. In 9th *International Conference on Artificial Intelligence*, pages 15–19. 1985.

[Pollack 82] Pollack, Martha E., Hirschberg, Julia, and Webber, Bonnie. User Participation in the Reasoning Processes of Expert Systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 358–361. 1982. A longer version of this paper appears as Technical Report MS-CIS-82-9, Department of Computer and Information Science, University of Pennsylvania.

[Pollack 85] Pollack, Martha E. *Goal Inference in Expert Systems.* Technical Report MS-CIS-84-07, Department of Computer and Information Science, University of Pennsylvania, 1985.

[Reiser 85] Reiser, Brian J., Anderson, John R., and Farrell, Robert G. Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming. In 9th *International Conference on Artificial Intelligence*, pages 8–14. 1985.

[Reiter 80] Reiter, Raymond. A Logic for Default Reasoning. *Artificial Intelligence* 13(1):81–132, 1980.

[Rich 79] Rich, Elaine. User Modelling Via Stereotypes. *Cognitive Science* 3:329–354, 1979.

[Schmolze 83] Schmolze, J. G. and Israel, D. *KL-ONE: Semantics and Classification.* Technical Report 5421, Bolt, Beranek and Newman, 1983.

[Shrager 82] Shrager, J. and Finin, Tim. *An Expert System that Volunteers Advice.* Technical Report MS-CIS-82-15, Department of Computer and Information Science, University of Pennsylvania, 1982.

[Sidner 81] Sidner, Candace L. and Israel, David J. Recognizing Intended Meaning and Speakers' Plans. In 7th *International Conference on Artificial Intelligence*, pages 203–208. 1981.

[Sleeman 82] Sleeman, D. Assessing Aspects of Competence in Basic Algebra. In Sleeman, D. and Brown, J. S. (editors), *Intelligent Tutoring Systems*, pages 185–200. Academic Press, New York, 1982.

[SparckJones 84] Sparck-Jones, Karen. *User Models and Expert Systems.* Technical Report 61, Computer Laboratory, University of Cambridge, Cambridge, England, 1984.

[Swartout 83] Swartout, William R. XPLAIN: A System for Creating and Explaining Expert Consulting Programs. *Artificial Intelligence* 21:285–325, 1983.

[Wahlster 86] Wahlster, W. and Kobsa, Alfred. Dialog-Based User Models. *Proceedings of the IEEE* forthcoming, 1986.

[Wilensky 86] Wilensky, R., Mayfield, J., Albert, A., Chin, D. N., Cox, C., Luria, M., Martin, J., and Wu, D. *UC - A Progress Report.* Technical Report UCB/CSD 87/303, Department of EECS, University of California, Berkeley, 1986.

[Zissos 85] Zissos, Adrian Y. and Witten, Ian H. User Modelling for a Computer Coach: A Case Study. *International Journal of Man-Machine Studies* 23:729–750, 1985.