Logic-Based Representation and Reasoning for User Modeling Shell Systems

Dissertation

vorgelegt zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

des Fachbereichs Mathematik und Informatik
der Universität – Gesamthochschule – Essen

von

Wolfgang Pohl

aus Bonn

Acknowledgements

At first sight, a dissertation is the work of a single person. To some extent, this is true. The main innovative ideas and in particular the detailed preparation and writing of the dissertation, not to forget about software programming, must be done by one individual in order to be acceptable as his own achievement. However, only very few, if any, scientific work is done in isolation and without any contribution, assistance, or support; and for this work, this is certainly not the case. Therefore, I would like to express my gratitude towards quite a large number of people, who more or less directly and more or less obviously have contributed to this work.

This dissertation presents work that has its origins in the BGP-MS project, which was funded by the German Science Foundation (DFG). This project had been proposed by Professor Alfred Kobsa, who gave me the opportunity to work on BGP-MS for five years at the University of Konstanz and the University of Essen, and who afterwards let me prepare this dissertation on a doctoral position in his research group at GMD Birlinghoven. He provided initial ideas and the background for this work. Last but not least, he supervised my Ph.D. work, so that I am very much indebted to him. I also have to and want to thank Professor Rainer Unland from the University of Essen, who very willingly accepted to be my second supervisor.

During my work on the BGP-MS project, I was privileged to cooperate and exchange ideas with many people in Konstanz, Essen, and at GMD. First of all, I want to mention Jörg Höhle, who was colleague and partner during the BGP-MS days. This work and I have significantly benefitted from his many talents, particularly from his mastery in programming and system administration, and from his ability to detect the little bugs and errors that are around in thoughts, words, and actions. There is a large number of other colleagues and students whom I have to thank for discussions, questions, and other contributions; the following list certainly does not mention all of them: Josef Fink, Rainer Hammwöhner, Detlef Küpper, Vito Longo, Andreas Nill, Holger Schauer, Jörg Schreck, Christoph Thomas, and Hans Zimmermann. In particular, Alexander Sigel gave especially valuable comments on a draft version of this dissertation. Special thanks go to Hans-Jürgen Ohlbach, who introduced me to the secrets of modal logic and who invented most of the methods that have become used in BGP-MS for processing modal logic with first-order logic theorem provers.

Of course, to get a dissertation done is not only a question of science. Lots of mental support, friendship, and love have been the indispensable prerequisites for the preparation of this work. A special promoter and friend has been Christine Harms – thank you a lot! Thanks also to Erich Vorwerk, who kept on pushing me forward. Similarly, I want to thank all other friends that showed interest in my work and encouraged me to proceed.

Unlimited support in all respects has always been provided by my family, who thus contributed a lot to this work. In particular, I am very grateful to my parents, who never stopped promoting my talents. Above all, they taught me to love music and arts, which to me has always been an important compensation of working in computer science. Now I will not let pass the chance to give something back, and so I dedicate this work to them.

Finally, I want to thank Andrea and little Clara for being there and with me.

Abstract

Interactive systems that attempt to adapt their behavior to the individual characteristics of their users need a user model, i.e., a source of information which contains assumptions on those aspects of a user that might be relevant for adaptation. In addition, in case of sufficiently complex user models, specific user modeling systems become necessary. One of the core parts of any such user modeling system is a mechanism for representing the user model and for reasoning about the assumptions about the user that are held in the model.

In the last decade of user modeling research, a number of user modeling shell systems has been built. Their purpose is to provide developers of user modeling systems with application-independent user modeling mechanisms, among them representation and reasoning facilities which ideally serve the needs of as many user modeling systems as possible.

User model representation is a knowledge representation task. In this work, we argue that in case of standard assumptions about the user, two aspects need to be represented, namely the assumption type (e.g., assumptions about user beliefs, assumption about user goals, etc.) and the assumption content (i.e., what a belief, goal, etc. is about). It will be shown that, like for knowledge representation in general, logic-based formalisms and methods have been heavily utilized in user modeling systems. Among them, there are mainly two competing representation approaches that explicitly deal with assumption type information and, in addition, are generic and powerful enough to become used in user modeling shell systems. One of them is the modal logic approach, which mainly claims that the user model should be represented using one single powerful formalism (a modal logic), which in a shell would possibly satisfy most representation needs. Such a logic offers merely unlimited representation and reasoning possibilities and can be equipped with a formal semantics. The other approach is a more pragmatical one: It represents assumption type information by using distinct partial knowledge bases, so-called partitions, to distinguish between assumption contents of different types. By employing distinct representation means for assumption types and contents, the partition approach can flexibly deal with both type and content information in specialized ways. However, it cannot mix both aspects as freely as modal logic so that its expressive power is limited. Moreover, it has been criticized for a lack of formal semantics.

In this work, we present the Assumption Type Representation (AsTRa in brief) framework for user model representation and reasoning, which is aimed at getting used in user modeling shell systems. It combines the mentioned approaches for obtaining both the power of modal logic and the flexibility of partitions. As the name suggests, the core elements of AsTRa are assumption types, which essentially are partition-like partial knowledge bases. Within assumption types, assumption contents can be maintained using an arbitrary number of logic-based formalisms. This basic part of the framework can be formalized and characterized

in terms of modal logic, thus inheriting the formal semantics of modal logic. On this foundation, it became possible to add extended modal logic representation facilities to AsTRa, the inference processes of which can take contents of assumption types into account. Since modal logic is only an option within AsTRa, the framework offers both the power of modal logic and the flexibility of a partition-like two-level representation of assumption type and assumption content information. As a side effect of the correlation between assumption types and modal logic, negative assumption types are introduced that offer a specialized way of handling negative assumptions (about what the user does not believe, want, etc.), the inference mechanisms of which are based upon modal logic reasoning. Altogether, the AsTRa inference procedures are offered at five different levels of reasoning, ranging from database-like user model access (i.e., no reasoning at all) to full modal logic reasoning, including processing of modal axioms.

The AsTRa framework has been implemented in the user modeling shell system BGP-MS. In this work, we describe this implementation and the mechanisms that we developed for realizing the integration of modal logic and partition-based representation in BGP-MS. The actual use of the resulting representation and reasoning facilities is illustrated in detail. Moreover, further components of BGP-MS that are in correlation to these facilities are briefly documented. Finally, related systems, mainly other user modeling shells, are discussed and compared with BGP-MS.

In sum, this work presents a promising progress in logic-based user model representation and reasoning. It reconciles the two previously predominant and seemingly competitive approaches in this area, thus bringing the advantages of both approaches together. The formal characteristics of the new approach are pointed out, and its feasibility is proven by implementation. From a methodological point of view, core aspects of user model representation and reasoning have been worked out. Extensions of this core are possible to deal with aspects of user modeling that go beyond the abilities of logic-based methods, like uncertainty handling and the use of inductive reasoning capabilities. Therefore, this work is relevant to a general user modeling audience. In particular, we invite everyone to adopt our terminology for talking about user model representation and reasoning. Thus, better mutual understanding within and, hence, faster progress of the user modeling research area may be promoted.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In their daily lives, people interact with a great number of artifacts for a wide range of purposes. Among the most powerful of those artifacts are computer systems, i.e. combinations of hardware and software components that constitute a system with a usually pre-determined behavior. The idea that computer systems or other artifacts, which human beings employ and interact with, could *adapt* to the characteristics of their human users has attracted a great number of researchers for a long time. Particularly in the Artificial Intelligence (AI) research area of natural language dialog systems, it was suggested that a system would be more cooperative if it could form its utterances according to knowledge it had acquired about the characteristics of each individual dialog partner [Wahlster and Kobsa, 1989]. But also in the research area of Human-Computer Interaction (HCI), notions like adaptivity and adaptability were investigated [Schwab, 1989; Oppermann, 1994]. In general, if a computer system is meant to be utilized by a diversity of users, its users will show different problems and needs as well as different degrees of familiarity or proficiency with the system. With the increasing complexity of the tasks that are supported by computer systems and of the information that is provided by computer-based information systems, also the need for system assistance and guidance increases. However, in order to assist users appropriately, systems must take information about the individual characteristics of each user into account.

The process of gathering information about the users of computer systems and of making this information available to systems which will exploit it to adapt their behavior or the information they provide to the specific requirements of individual users has been termed *user modeling*. Then, a *user model* is a source of information, which contains assumptions about those aspects of a user that might be relevant for behavior or information adaptation. The term has also been used in a different way in Human-Computer Interaction (HCI) research: there, it often refers to an a-priori model of the users of a computer system that the system designer has in mind, or to the assumed models that users will probably develop of the system and the tasks they can perform using the system (see, e.g., [Norman and Draper, 1986]). We will stick to the more AI-related notion of a user model as an explicitly maintained knowledge base, which is augmented and exploited during the interaction between system and user.

In about twenty years of user modeling research, a wide range of user modeling techniques and mechanisms have been developed. However, a common set of user modeling tasks has evolved that almost every user modeling system has to accomplish. User modeling systems need mechanisms that (cf. [Kobsa and Pohl, 1995])

- form assumptions about the user based on her interaction with the computer system,
- represent and store these assumptions in a knowledge base,
- infer additional assumptions from already held assumptions,
- handle inconsistencies between assumptions, and
- supply the system with information about the currently held assumptions.

Another important user modeling task is to provide controlled access to user models, i.e. to let the user access her model for purposes of inspection and perhaps modification, but to not allow unauthorized access to and abuse of assumptions about the user.

The development of such mechanisms can be quite costly. Therefore, considerable effort has been spent on building application-independent *user modeling shell systems*, which provide implementations of basic, application-independent user modeling mechanisms to the developers of user modeling systems [Finin, 1989; Brajnik and Tasso, 1994; Ikeda and Mizoguchi, 1994; Kobsa and Pohl, 1995; Kay, 1995; Orwant, 1995; Paiva and Self, 1995]. One of the core parts of user modeling systems is a mechanism for representing the user model and for reasoning about the assumptions that are held about a user. So, any user modeling shell should offer representation and reasoning facilities which can serve the needs of as many computer systems as possible.

The primary goal of this work was to develop an application-independent representation and reasoning system that is suitable for a wide range of user modeling applications and therefore is appropriate for getting implemented in user modeling shell systems. Essentially, it pursues a traditional approach to representing and reasoning about knowledge: it employs logic-based techniques. It will be shown that many user modeling systems have used logic-based, symbolic representation and reasoning methods. The problems of logic-based techniques, like the lack of possibilities for representing uncertainty and the difficulty of dealing with the nonmonotonic evolution of user models will be discussed in Chapter 9. It will be sketched briefly that in principle these problems may be tackled with extensions to logic-based user modeling methods. These extensions, however, are not within the scope of this thesis.

The novel contribution of this thesis is a framework for logic-based user model representation and reasoning, which was designed particularly for user modeling shell systems. It was significantly influenced by the experiences that the author made concerning both development and use of the user modeling shell system BGP-MS [Kobsa and Pohl, 1995]. This framework attempts to cope with the desire to build tools for a wide range of user modeling tasks and problems. The basic idea is to provide a variety of representation and reasoning techniques in a shell system, which a user modeling application flexibly can choose from, according to its specific needs. In addition, all available methods can be used in an integrated fashion by applications that have very sophisticated demands and need a powerful representation and reasoning system.

1.1 Logic-Based User Modeling

1.1.1 User Models and Knowledge Bases

Logic-based knowledge representation requires the use of a representation *formalism*. Such a formalism is first characterized by its *syntax*; syntax allows the construction of more or less complex symbolic expressions (often also called formulas), which are meant to describe an aspect of a state of affairs. Furthermore, a logic-based formalism provides a *semantics* that is based on a notion of truth; it allows one to determine whether a symbolic expression holds true in a world that gives a meaning to the symbols in the expression. Hence, logic-based formalisms can be used to represent a state of affairs which can be characterized by a set of expressions that are all true in this state.

In general, a user model represents a state of a software system with respect to the assumptions the system makes about the user. Logic-based user model representation then means to describe this state with expressions of a logic-based knowledge representation formalism. The user model is regarded as a set of expressions, which symbolize the assumptions of the system that are true in the current system state. Following this idea, a logic-based user model is a (logic-based) knowledge base.

But there is more to a logic-based knowledge representation formalism than just the formation of symbolic expressions that can be interpreted meaningfully. In a logic-based formalism, the truth of an expression can be related with the truth other expressions that share symbols with the first one. In general, if there is a set of explicitly constructed expressions that are true, then there will be other expressions that must be true, too. A logic-based knowledge representation formalism provides *reasoning mechanisms* that permit one to automatically determine such implicit conclusions from explicitly represented knowledge. This property is interesting for user modeling systems; explicit information about users is often rare, so that reasoning mechanisms can be employed to acquire more information by inferring implicit user model contents. For instance, let a logic-based user model contain expressions representing the (assumed) user beliefs that the printer in his office (called “lj1”) is of type “laserjet”, and that all laserjets are laser printers. A conclusion from these user model contents would be that the user (implicitly) believes that lj1 is a laser printer. The process of inferring the implicit contents of a knowledge base is also called *deduction*[1]. So, in logic-based user modeling, a user model is regarded as a *deductive knowledge base*.

However, in most cases assumptions about the user alone do not allow one to make interesting deductions. In general, for logic-based inference[2] of implicit knowledge from a given knowledge base, specific expressions are required that represent truth relationships between (other) expressions. In case of user modeling, often expressions are needed for inferences which themselves are not assumptions about the user but represent truth relationships between possible assumptions about the user. For instance (considering printers and printing again), the following rule could be formulated in a logical formalism: “If the user is assumed to believe that lj1 will print some PostScript document, then she can also be assumed to believe that lj1 is a PostScript-capable printer”. This rule is not an assumption about the user, but an expression on a meta-level that can be used by an inference mechanism to derive assumptions about the user.

Similarly, in many user modeling systems there are so-called acquisition rules that determine what can be assumed about the user based (1) on her behavior in her interaction with a system, (2) on the currently held assumptions about her, or (3) on both. Particularly the latter two kinds of rules are candidates for getting represented in a logic-based knowledge representation formalism, which is already used to represent assumptions about the user. In a strict sense, acquisition rules are not part of the user model, since they do not represent an assumption about the user. Therefore, logic-based user modeling does not deal with the user model only, but with user modeling knowledge (which includes the user model itself) in general. That is, a deductive *user modeling knowledge base* (UMKB) is maintained that contains both assumptions about the user and other knowledge that will help to infer implicit assumptions.

[1]Reasoning based on the principles of both induction (see, e.g., [Webb and Kuzmycz, 1996; Yoshida and Motoda, 1996]) and abduction (see, e.g., [Appelt and Pollack, 1992; Stein *et al.*, 1997]) may also be used to infer assumptions about the user. However, these reasoning principles are not truth-preserving, i.e., they do not detect assumptions that are implicit in a strictly logical sense. Hence, we do not regard inductive and abductive reasoning as elements of truly logic-based user modeling. See Section 9.3 for a discussion of reasoning principles and user modeling.

[2]Note that the concepts “inference” and “reasoning” are very closely related. In this work, we mainly use reasoning as a more general notion, while inference tends to be used when we have single reasoning activities in mind.

1.1.2 User Modeling Shells and Deductive Knowledge Bases

According to the foregoing considerations, a user modeling shell system that supports a logic-based user modeling approach will naturally provide deductive knowledge base mechanisms that can be employed by user modeling applications to maintain their user modeling knowledge bases. Of course, this is not a new idea, particularly not for user modeling shell systems. In [Finin, 1989], the shell GUMS is said to follow the "paradigm of user model as deductive database" (p.418). In this thesis, the term "deductive database" is replaced by "deductive knowledge base". The reason is that the former term has acquired a dedicated technical meaning which was established in the area of logic programming [Minker, 1988].

Now, we will take a closer and slightly more formal look at deductive knowledge bases to get a better impression, what logic-based representation and reasoning mechanisms in a user modeling shell system will look like. Deductive knowledge bases are somewhat similar to classical databases. They contain a set of entries (here: formulas); there must be a mechanism to add entries to the knowledge base (**tell**), and a query mechanism to retrieve entries (**ask**). In contrast to classical databases, however, a knowledge base may contain implicit entries, which are not explicitly stored. This is due to the notion of (semantic) *entailment* in logical formalisms. If all formulas of a given knowledge base KB are true or valid, then other formulas can be found which must be valid, too. For such a formula ϕ, it is said that KB entails ϕ, short $\mathrm{KB} \models \phi$. As far as logical truth is concerned, there is no difference between explicit entries and formulas that are entailed by these entries. So, a fundamental characteristic of a deductive knowledge base is that retrieval mechanisms permit access to both explicit and implicit entries. The query function **ask** receives a formula ϕ and a knowledge base KB as arguments and gives a positive response if and only if ϕ is valid according to KB: $\mathbf{ask}(\mathrm{KB}, \phi) = \text{true iff} \mathrm{KB} \models \phi$. Thus, the **ask** function determines what follows from the entries that were **tell**ed (cf. [Russell and Norvig, 1995]).

In order to realize the **ask** function, a calculus is needed that permits the automated deduction or inference of entailed formulas for a given knowledge base KB. A calculus itself establishes a derivation relation $\vdash$; the calculus is useful if this relation is equivalent to the $\models$ relation: $\mathrm{KB} \vdash \phi \iff \mathrm{KB} \models \phi$, i.e., if it is sound (it does not derive formulas that are not entailed) and complete (it derives all entailed formulas). Given such a calculus, the **ask** function can now be defined as $\mathbf{ask}(\mathrm{KB}, \phi) = \text{yes iff} \mathrm{KB} \vdash \phi$. Note that for a computer implementation I of a calculus, the resulting $\vdash_I$ relation normally cannot be proven to equivalently implement $\vdash$ and hence to equivalently implement $\models$, even if $\vdash$ is sound and complete. However, this is a theoretical problem which does not prevent computer implementations of calculi from being used successfully.

Calculus implementations are often referred to as inference components or inference engines of logic-based knowledge representation formalisms. In combination with deductive knowledge bases, they are not only used by an **ask** function to verify derivability, but their deductive capabilities can also be employed to provide other inference services. Not only can derivations be made on the occasion of a query; also, the addition of a new entry may lead to new derivations, which can be computed immediately. Moreover, the consistency of a new entry with the existing knowledge base can be checked with the help of an inference engine.

A user modeling shell system will implement such inference services in a generic fashion. Then, the developer of a user modeling application (the user model developer) can employ these deductive knowledge base mechanisms for setting up a-priori user modeling knowledge, e.g. acquisition rules or domain knowledge that may be relevant to the user modeling process, for entering assumptions into the user model, and for retrieving explicit as well as implicit assumptions that are needed to determine the adaptive behavior of the application.

1.2 Other User Modeling Approaches

In the previous section, we said that with a logic-based formalism, assumptions about the user are represented by assigning the values "true" or "false" to expressions of the formalism. Then, expressions that are valued true are assumptions about the user. Inference mechanisms are able to derive further true expressions—and hence further assumptions about the user—from a given set of assumptions, based on complex expressions that denote truth relationships between simpler expressions. So, in logic-based user modeling, it is possible to tell whether an expression (perhaps implicitly) is an assumption about the user or not.

In many user modeling systems in the literature, the state of the system with respect to the assumptions it makes about the user has been represented using a finer-grained set of values. Typically, a range of numeric or symbolic values has been employed to assign to knowledge items a *degree of evidence* (we will also use the term *graduation*) of their being an assumption about the user. In general, it is said that these evidence values represent the uncertainty of the system whether something can really be assumed about the user or not. In Section 2.2, we will see that evidence values can be used for different kinds of graduation. Examples of user modeling systems using an *evidence-based* approach are

- UMFE [Sleeman, 1985], which assigns the values known, not-known, and no-information to concepts in the domain of bacteriology in order to express the degree of the system's assessment of the user's knowledge of these concepts;

- the system of Jennings and Higuchi [Jennings and Higuchi, 1993], which employs nodes in a neural network to represent content features of news articles. The activity value of each node expresses the assumed strength of a user's interest in the corresponding feature.

- EPI-UMOD [De Rosis *et al.*, 1992], in which assumptions about user knowledge of concepts in the area of epidemiology are modeled with Bayesian networks. Each network node corresponds to a statement about the user knowing a certain concept. The probability value of a node is a measure of the probability that this statement is true, i.e., that the user knows the concept under consideration.

In evidence-based systems, inferences can be made by computing the graduation value of a representation item from the values of other items. In UMFE, there are rules that determine this *propagation of evidence*. When network representation methods like neural networks and Bayesian networks are employed, propagation is determined by network links and their weights or conditional probabilities respectively.

At first sight, logic-based user modeling might be regarded as a special case of evidence-based user modeling: The evidence values true and false are assigned to logical expressions, and these evidences are propagated by inferences based on specific truth-relating expressions. However, there is an interesting property of logic-based techniques, which many evidence-based systems do not have: Inferences in logic-based systems may dynamically form new expressions, while the set of knowledge items (e.g., the nodes of a neural or Bayesian network) is fixed in many evidence-based systems[3]. Of course, logic-based systems do not *invent* new assumptions about the user, but a possibly infinite set of assumption expressions can be formed based on a vocabulary, which itself may be extended dynamically. For example, a logic-based system that has to deal with a dynamic set of objects will typically create a new designator when a new object needs to be denoted in a logical expression.

However, dynamic construction of user model contents also is an issue when using evidence-based techniques. There is some work that has tackled this problem. Jennings and Higuchi dynamically add a node to a neural network when a novel content feature appears in a news article, and they delete nodes the activity values of which fall below a certain threshold. For the natural language dialog system PRACMA [Jameson *et al.*, 1995], Jameson presents a very interesting method for dynamic generation of Bayesian networks [Jameson, 1995]. In order to be able to make predictions about user knowledge, he employs the inference traces of a logic-based system, which provide possible explanations of the user's knowing some fact. The traces are transformed into Bayesian networks, which are to handle the uncertainty in these explanations.

For an overview of systems that use numerical uncertainty management for user modeling, see [Jameson, 1996]. Other examples of the application of neural

[3]In recent work, dynamic Bayesian networks have been employed for better taking temporally varying user characteristics into account [Schäfer and Weyrath, 1997]. In these networks, nodes may get replicated for distinct treatment in different time slices, but no essentially new nodes are generated.

networks are described in [Lindner and Bodendorf, 1993; Gori *et al.*, 1997; Ye, 1997].

In this thesis, we focus on the use of logic-based methods for user modeling. However, in Sections 2.2 and 9.2 we will discuss how evidence or graduation values can be related to the approach pursued in this thesis.

1.3 What Is in a User Model? Assumption Types and Contents

Throughout this thesis, it will become clear that logical formalisms are used in mainly two different ways in order to form a user model as a deductive knowledge base. User models typically contain assumptions about the beliefs, the goals, the preferences, or other attitudes of the user. Taking user beliefs as example, a user model content could be, e.g.,

(The system assumes that / It is assumed that)
The user believes that
UNIX is an operating system for workstations.

Now, the first approach to represent this user model content is to formulate the whole sentence (including the part in parentheses or not) as one logical expression. However, if the user model is only to contain assumptions about user beliefs, it is sufficient to have an expression for

UNIX is an operating system for workstations.

in the user model in order to represent the first sentence.

Many user modeling systems follow the second approach. It differs from the first one that the *assumption type* (in the example: assumption about the beliefs of the user) is not represented in the logical expression, but by regarding the user model as container for assumptions of one kind, which is used to store the *assumption content* only. Particularly if only one assumption type is to be modeled, the container approach will obviously avoid redundancy and allow the use of a less complex representation formalism. If the user model consists of assumptions of different types, then several containers, one for each type, can be employed. Then, it is necessary to label the containers such that assumption type information is not lost. The necessary precondition for employing containers, also called partial knowledge bases or *partitions* (see next section), is that an assumption about the user can be split up into assumption type and content, so that it can be written as

assumption type : *assumption content*

The above-mentioned example assumption can be written as

assumption about user beliefs :
UNIX is an operating system for workstations.

The distinction between assumption type and assumption content will be important throughout the whole thesis. The above AT:ac notation for user model contents with assumption type AT and assumption content ac will be used frequently. Such expressions will be referred to as *assumption type expressions* or as *type-internal expressions*.

1.4 User Modeling Shells: Power and Flexibility?

No tool can be generic enough to satisfy all demands; this is also the case for user modeling shell systems. The range of techniques that have been utilized in user modeling systems for user model representation and reasoning is quite wide, and no user modeling shell or tool system can be expected to cover all possible needs. However, a logic-based user modeling shell system may have the following two goals concerning user model representation and reasoning:

- Representation and reasoning mechanisms should be powerful. I.e., the expressive power and reasoning capabilities of the formalism a shell provides should satisfy sophisticated demands in order to be applicable to a number of user modeling problems which is as large as possible.
- Representation and reasoning mechanisms should be flexible. I.e., in order to be useful and appropriate to systems with lesser demands, the shell should offer the possibility to choose less powerful but perhaps more efficient variants of its representation and reasoning techniques.

We will briefly sketch two approaches to user model representation that both provide quite powerful representation mechanisms. The first one has an advantage concerning expressive power, while the second one is preferable because of its potential for flexibility. This thesis attempts to integrate both approaches into a powerful and flexible representation and reasoning framework.

The Modal Logic Approach

In order to achieve representation and reasoning power, a single powerful formalism can be used. Since user modeling mostly deals with beliefs and related epistemic notions, epistemic logic comes to mind. Modeling attitudes like knowledge or belief with logic has its origins in philosophy. Lewis was the first to introduce and axiomatize modal, non-truth-functional operators that added the notions “necessary” and “possible” to logical formulas [Lewis and Langford, 1932].

The use of related operators in logics for modeling informational attitudes like knowledge and belief started with [Hintikka, 1962]. Hintikka also proposed the possible worlds semantics for logics of knowledge and belief, which was refined and improved by Kripke [Kripke, 1963] to the form that is used with all kinds of modal logics until today.

In most cases, researchers were interested in modeling knowledge and belief or other attitudes of a single agent. However, in the areas of natural language processing and distributed problem-solving the need to model beliefs of one agent (a natural language system or a problem solver) about the attitudes of another agent (a dialog partner or another problem solver) arose. Modal logic is applicable also in this case, if different operators are introduced for each agent. The pure use of modal logic in the area of user modeling was advocated in [Allgayer *et al.*, 1992]. In [Hustadt, 1994; Hustadt, 1995], Hustadt presents an extension of description logics with generic modal operators, which can be parameterized with modality and agent identifiers, thus obtaining a multi-modal, multi-agent logic for user modeling. In Section 3.4, this work and several other examples of the use of modal logic for user modeling will be described.

In general, modal logics permit to introduce all kinds of operators to characterize propositions. Thus, in principle, various modalities or attitudes that are relevant to user modeling like goals, preferences, intentions, etc. can be expressed with modal logic. However, for a user modeling shell system there is a severe disadvantage in providing one single powerful formalism only. User modeling applications with special, but less sophisticated demands are forced to employ a formalism which is not tailored to their needs and therefore perhaps unnecessarily inefficient.

The Partition Approach

Above, we observed that assumptions about the user consist of two main elements, namely assumption type and assumption content. In order to achieve representational flexibility, this dichotomy of user model contents can be represented explicitly. If a user modeling shell system provides a two-level representation with a separation of type and content, the representation tasks of each level can be handled in their own specific way. On each level, less sophisticated mechanisms can be employed which need not cover both aspects of a user model content.

The use of modal logic for an integrated representation of type and content indicates that type information assigns a status to content information. Most representation formalisms are not capable of expressing such status information, but can nevertheless be employed for assumption contents. Therefore, separate handling of assumption type and content opens the door for all such formalisms on the content level. Thus, a two-level representation system may offer several content formalisms, which is useful for mainly the following reasons: Quite different assumption types are possible in a user modeling system, the contents of which may need to be represented in specific and suitable ways. Even if only one type is concerned, there may be different kinds of contents, so that the employment of specialized formalisms would be beneficial again. Hence, a user modeling shell with a two-level representation should provide more than one content formalism in order to offer representation alternatives, or to permit the parallel use of several formalisms for specialized representation of different kinds of contents in one knowledge base.

A two-level representation of assumption type and content is accomplished in the *partition approach*, which was first employed by Cohen to extend semantic networks with a possibility to distinguish different assumption types [Cohen, 1978]. Cohen uses partitions as "spaces" to collect knowledge items. Each space (i.e., partition) is characterized by a belief and goal nesting. E.g., an SB partition stores all items that are considered System Beliefs, SBUW contains System Beliefs about what the User Wants, etc. Partitions are nested; a knowledge base typically contains a "nesting hierarchy" of partitions with SB as the outermost partition (see Section 2.3.1 for details). Similar to Cohen's approach, nested partitions were discussed by Ballim [Ballim and Wilks, 1991b; Ballim, 1992] to represent belief nestings regarding different agents (cf. Section 2.3.4). Kobsa used partitions for representing user models in the system VIE-DPM [Kobsa, 1985], the user modeling component of the natural language processing system VIE-LANG (cf. Section 2.3.2). In VIE-DPM, partitions are seen as contexts that contain beliefs or goals at one level of nesting. In this system, partitions do not contain inner partitions; they are separate entities, which are however linked to other partitions that represent a deeper nesting of beliefs and goals. Later, in early work on the user modeling shell system BGP-MS [Kobsa, 1990], partition nesting became less important. Instead, partition inheritance was introduced to allow, e.g., common or mutual beliefs of dialog partners to be stored in distinct partitions, which inherit their contents to other partitions that store the beliefs of each partner (cf. Section 2.3.3). Essentially, this later partition mechanism was retained in BGP-MS and has been used as basic means for the implementation of the representation framework that is proposed in this thesis.

A general property of all work within the partition approach is that, within a global user modeling knowledge base, partial knowledge bases are distinguished, each of which stores the contents of one of the several types of knowledge that are to be modeled in a system. Thus, the partition approach permits two-level representation. Partial knowledge bases convey assumption type information on the first level, and they store assumption contents, which constitute the second level.

However, there is a severe disadvantage of any partition mechanism: It can be used for the representation of assumptions about the user, but it is less suitable for meta-level expressions like acquisition rules, which enable reasoning procedures to make inferences. If inference rules or similar expressions concern assumptions of one type only, then it is possible to handle them within the partial knowledge base that stores the assumption contents of that type. For instance, the example rule about PostScript printing presented on page 4 only deals with assumptions about user beliefs: "if the user is assumed to believe that lj1 will print some PostScript document, then he can also be assumed to believe that lj1 is a PostScript-capable printer". It could be represented as an assumption of type "system beliefs about user beliefs" (SBUB), with the assumption content being the rule "if lj1 prints some PostScript document then it is a PostScript-capable printer". Similar to the previous version, this rule would permit to infer SBUB assumptions about the PostScript-capability of lj1. However, it is impossible to specify relationships between assumptions of different types within a UMKB the contents of which are organized into type-specific partitions only. Hence, there is a need for mechanisms that are able to integrate the assumption contents of different types. Such an integration is a difficult problem, particularly if different formalisms are used within assumption type partitions.

An Integrative Approach

In order to integrate different methods, a unified view on the UMKB must be possible. Above, we discussed modal logics as a means for an integrated representation of user modeling knowledge, but criticized its lack of flexibility. The use of two separate levels of representation for assumption types and contents, like in the partition approach, provides more flexibility, but is problematic as far as expressive power is concerned.

In this thesis, a framework for *assumption type representation* (AsTRa in brief) is developed that specifies its own notion of a deductive user modeling knowledge base. In AsTRa, a UMKB consists of a number of partial knowledge bases, which may contain expressions of an arbitrary number of formalisms for content representation. So, the flexibility of a two-level representation is available. However, the AsTRa specification permits the characterization of UMKB contents in terms of modal logics. On that foundation, the partial knowledge base approach can be extended with modal logic representation and reasoning means in an integrative way. [Hustadt, 1995] briefly reviews the partition approach and mainly criticizes its lack of semantics and that reasoning is confined to one partition. It is true that the AsTRa framework basically is in line with the partition approach, but because of its accordance with modal logic it has a semantics, and the modal logic extension permits reasoning beyond partial knowledge bases.

1.5 Organization of This Work

This work consists of three parts, with two main parts and one concluding discussion part.

Part I will give an extensive overview of logic-based representation and reasoning techniques as they have been utilized in user modeling systems, and introduces as well as discusses the relevant issues[4]. In the next chapter, the notion of assumption types is examined in more detail. Based on the user modeling literature, a range of possible types of assumptions about the user is determined. In addition, user modeling systems are reviewed that employ a distinct representation level for assumption type information. These systems established the partition approach to user model representation. The ideas of the AsTRa framework will be related to these systems and discussed in more detail.

In the subsequent two chapters (3 and 4), the user modeling literature is reviewed for user modeling systems and approaches that make use of logic-based representation techniques. First, logic-based user model *representation* methods are described. It will turn out that logic-based formalisms of arbitrary complexity have been utilized for user modeling. The representation schemes of the investigated systems will be related to an assumption-type-oriented two-level representation. Thus, the usefulness of a user modeling shell system with a representation system of that kind can be demonstrated. Knowledge representation is always correlated to *reasoning*. So, second, several examples of systems that use logic-based reasoning and the corresponding inference procedures will be presented. This is to determine the kinds of inferences that are of interest to user modeling systems and therefore should be available in user modeling shell systems.

Part II presents a framework for user model representation and reasoning to be used in user modeling shells, and describes its implementation in the user modeling shell BGP-MS. Chapter 5 is the central chapter of this work. There, the AsTRa framework for logic-based user modeling is specified, which is based on the idea of representing assumption types as partial knowledge bases within a UMKB. Besides partial knowledge bases, UMKB access functions are the basic elements of an AsTRa system. They are based on the facilities that are provided by assumption content formalisms and provide an interface to the inference procedures of an AsTRa system. A fundamental relationship between basic AsTRa mechanisms and modal logic can be proven if all content formalisms satisfy certain conditions. On the basis of this relationship, the framework can be extended with modal logic representation and reasoning techniques. They are particularly useful for expressing inference rules that relate assumptions of different types, but also for negative assumptions, i.e., assumptions about what the user does not believe, want, prefer, etc. In accordance with the idea of flexibility, a specific kind of assumption types for representing negative assumptions is introduced. Such negative assumption types provide a simple representation mechanism along with specialized reasoning procedures that can be used instead of or in combination with complex modal logic methods.

The AsTRa framework has been implemented in the user modeling shell system

[4]The title of this first part is partly borrowed from Anthony Jameson [Jameson, 1996].

BGP-MS. Chapter 6 describes the representation and reasoning facilities of BGP-MS in AsTRa terms. BGP-MS offers the full range of AsTRa methods, including full modal logic reasoning. The conceptual representation system SB-ONE and first-order predicate calculus (FOPC) are the available content formalisms. FOPC reasoning was realized by using an automated theorem prover. It is shown that both formalisms are logic-based.

In Chapter 7, the actual use of the representation and reasoning facilities of BGP-MS is illustrated. Both phases of the employment of a user modeling shell system, i.e., the development time of a specific user modeling system and its run time, are covered by a coherent example. Moreover, additional components of BGP-MS, which are related to its representation and reasoning mechanisms, are briefly documented.

Part III concludes this work. Chapter 8 discusses related systems. Among them is a representation system that is not especially dedicated to user modeling, but is very similar to BGP-MS in its use of partition-like mechanisms and of multiple content formalisms. In addition, the most important user modeling shell systems are discussed, with a focus on user model representation and reasoning.

In the final Chapter 9, we summarize this work, point out shortcomings of logic-based user model representation, and discuss perspectives of (logic-based) user modeling systems and tools.

Part I

Logic-Based User Modeling: An Overview of Systems and Issues

Chapter 2

Assumption Type Representation

2.1 Types of Assumptions about the User

The representation component of a user modeling shell system should be general enough to allow the representation of a wide range of assumptions about the user, and of domain knowledge and further user modeling knowledge including inference rules for model acquisition. This section attempts to make a statement about which types of assumptions may occur in user models. Ideally, a user modeling shell will be able to deal with as many of these types as possible and may additionally handle domain and user modeling knowledge of user modeling systems.

There have been several reviews of the field of user modeling containing statements about the range of user model contents. On a surface level, there is a high degree of agreement in the user modeling community. [Kass and Finin, 1988] regard goals and plans, capabilities, attitudes and preferences, knowledge and beliefs of users as being important for user modeling in natural-language (NL) systems. In [Wahlster and Kobsa, 1989], the need for an "explicit model of the user's beliefs, goals, and plans" is stressed for NL systems that are supposed to show cooperative behavior. A more recent review [McTear, 1993] says that goals and plans, capabilities, preferences, and beliefs and knowledge of the user are often modeled. From these statements it may be concluded that the representation of a flat list of assumption types seems to be sufficient for a user modeling shell. The set of supported types should at least include system assumptions (beliefs) about user beliefs, goals and plans.

But particularly if beliefs are concerned, there can be much more. Everything in a user model can be regarded as a belief of the user modeling system. So, when user beliefs are modeled, there are two "believers" to be considered, namely user and system. In the general case of modeling beliefs of agents that communicate and perhaps cooperate with each other, a great deal of research has been spent on nested beliefs, i.e. beliefs of an agent A about the beliefs of an agent B, beliefs of A about beliefs of B about beliefs of A, etc. (see [Taylor *et al.*, 1996] for references to relevant work).

Also in user modeling, nested beliefs have been considered. In his taxonomy of beliefs and goals for user modeling, Kobsa calls them "complex beliefs", and he also considers nesting in case of beliefs about goals [Kobsa, 1989]. So, a lot of additional assumption types can be considered for user modeling. In order to name these types, we will use a notation that follows [Cohen, 1978] and was similarly utilized in quite a number of papers [Allen and Miller, 1991; Kobsa, 1985; Kobsa, 1990; Kobsa and Pohl, 1995; Taylor *et al.*, 1996]. In this notation, assumption type labels consist of sequences of agent-modality pairs. In case of user modeling, standard agents are S (the system) and U (the user), but other agents are also possible. The most frequently used modality is B (for believes); in addition, W (for wants) is often employed to refer to the goals of agents. In principle, arbitrary modalities can be utilized in this kind of assumption type labeling, e.g. Pref for Preferences, I for intentions, etc. Using this scheme, SBUB denotes the class of assumptions (i.e., system beliefs) about user beliefs, and SBUW stands for assumptions about user goals. Assumption types involving nested beliefs are SBUBSBUB (assumptions about what the user believes the system to assume about her beliefs) or SBUBSBUW (assumptions about what the user believes the system to assume about her goals). Since in user modeling only assumptions (i.e., beliefs) of the system are considered, the first agent-modality pair in such a label will always be SB.

In the literature on belief modeling in natural-language dialog, special types of beliefs are considered that comprise an infinite number of other belief types. The most important notion is that of mutual belief. System assumptions about mutual beliefs of system and user comprise the assumption types SB, SBUB, SBUBSB, SBUBSBUB, etc.; SBMB can be used as label for this assumption type [Kobsa, 1990]. Kobsa criticizes that "the concept of mutual beliefs seems to be somewhat too narrow" [Kobsa, 1989, p.63]. So, he additionally introduces the notion of infinite-reflexive beliefs. This kind of beliefs deal not only with beliefs about beliefs, but also with beliefs in respect to other attitudes like goals or preferences. Mainly two kinds of infinite-reflexive beliefs exist, depending on to which agent the belief is ascribed. For instance, an infinite-reflexive belief of the system about a user goal g means that the system believes the user wants g (SBUW), that the system believes that the user believes the system belief about her goal (SBUBSBUW), etc. Using the M modality, these infinite list of assumption types can be summarized with SBMBUW. Analogously, infinite-reflexive beliefs of the system about user *beliefs* can be labeled SBMBUB. An infinite-reflexive belief of the user about a system belief is similarly composed of the assumption types UBSB, UBSBUBSB, etc., and can be labeled UBMBSB. However, in a user modeling scenario everything is a system belief, so that we arrive at SBUBMBSB, which comprises SBUBSB, SBUBSBUBSB, etc. It is easy to see that together with SB, the two infinite-reflexive types SBMBUB and SBUBMBSB summarize the mutual beliefs of system and user, SBMB (cf. [Kobsa, 1984; Kobsa, 1990]; see also Section 2.3.3).

[Taylor *et al.*, 1996] discuss belief modeling in cooperative dialog scenarios involving two agents. In particular, they investigate the necessary depth of nested belief models in cooperative dialog. In their terminology, a simulated agent, which in the case of user modeling is the system S as user modeling agent, has beliefs at the object level (SB) and at the first, second, etc. level of nesting concerning its beliefs and those of a dialog partner (SBUB, SBUBSB, etc., if the partner agent is named U). The most important result of their investigation is that a simulated agent need only represent beliefs at the object level and at the first two levels of nesting. However, in order to not exclude deeper nestings completely, they replace the second level of nesting with the notion of *residual mutual belief (RMB)*. A residual mutual belief comprises all nesting levels from the second onwards. With partner agent U, an RMB of the system (SRMB) is equivalent to SBUBSBMB in the above notation, which comprises SBUBSB, SBUBSBUB, etc. According to Taylor et al., an agent needs to distinguish three levels of beliefs, namely the agent's private beliefs (SB for agent S), the agent's beliefs about her partner's beliefs (SBUB), and the agent's beliefs about her partner's beliefs about what the agent believes to be mutually believed (SRMB, equivalent to SBUBSBMB). This result is interesting for the developers of user modeling systems that are to provide a belief model for cooperative scenarios. Unfortunately, [Taylor *et al.*, 1996] do not make any statement about beliefs involving other attitudes of agents like goals, intentions, or preferences.

An even more detailed "taxonomy of beliefs and goals for user models in dialog systems" is presented in [Kobsa, 1989]. Kobsa distinguishes between basic beliefs and goals on the one hand and complex beliefs and goals on the other hand. Basic beliefs and goals are beliefs and goals of one agent, which do not involve beliefs or goals of other agents. A further division of basic beliefs and goals into saturated and unsaturated beliefs and goals is based on properties of the assumption content. A saturated belief or goal concerns a fact that could be logically formalized without both free variables and existential quantification. Unsaturated beliefs and goals concern contents with an existential quantification, i.e., it is believed or wanted that there is an object with a specific property. Complex beliefs and goals are beliefs and goals about the basic and complex beliefs and goals of other agents. So, the taxonomy also includes infinite-reflexive beliefs concerning beliefs and goals. The assumption type representation framework that is presented in this thesis is based on a clear separation of assumption type and content. According to the intuition behind these notions, which was presented in Section 1.3, Kobsa's belief and goal classes partly refer to properties of assumption contents, so that they do not identify further assumption types in our sense of the term. However, they cover interesting kinds of beliefs and goals, like (in case of beliefs) "to believe that something is the case", "to be uncertain whether something is the case", "to know whether something is the case", and "to know the object x such that a predicate P holds for x". According to Kobsa, these kinds of beliefs and goals frequently occur in user-model-based dialog understanding and planning and must therefore be adequately representable in user models for such applications.

2.2 Assumption Types and Graduation

As Kobsa's above-mentioned notion of uncertainty indicates, it may be desirable to express the strength of an assumption or the degree of certainty or evidence that a system has in making an assumption about the user. Particularly in recent years, many user modeling systems employed some way of representing uncertainty, using a range of numeric or symbolic values for a *graduation* of assumptions (cf. also Section 1.2).

In principle, a graduation can involve both agents of the user modeling process, S and U. In order to make this clear, we will mainly focus on simple system assumptions about the user, i.e. all kinds of SBUX assumptions. First, the *strength of an assumption* may be graduated in order to express that the system believes to a certain degree that the user has a belief, a preference, etc. We will denote graduation with an asterisk (*), so that such assumptions can be characterized as SB*UX. Second, *assumptions about the strength* of the user's beliefs, preferences, etc. can be made. This means that a graduation concerning the user is expressed, but not a graduation concerning the system. The type of such assumptions can therefore be denoted by SBUX*. Both kinds of graduation can be mixed; e.g., Grundy [Rich, 1979] uses numerical values to express both the (assumed) strengths of user preferences and the certainty of the system about its assumptions. The difference between the two kinds of graduation corresponds to the use of defaults in Nested Theorist [van Arragon, 1991]. van Arragon distinguishes default assumptions of the system about the user from assumptions of the system about user defaults. Defaults are a means of expressing uncertainty without certainty values. So, since Nested Theorist only considers user beliefs, it uses the assumption types SB*UB and SBUB*. In deeper nested assumptions, it is possible to have a graduation at any level of nesting, so that in principle graduated assumption types like SB*UB*SB*UW* etc. are possible.

The focus of this thesis is the use of logic-based techniques for user model representation. In general, the use of a logical formalism for user model representation allows the distinction of two states, as far as a single assumption is concerned: The assumption is among the current assumptions about the user as (explicitly or implicitly) represented in the user model, or not. In logical terms: the assumption is entailed by the knowledge base that constitutes the user model or not. However, if the logical formalism used contains a negation operator, the semantic relation between assumptions and their negations can be employed to introduce at least a third state. Seen from a superficial point of view, "the user believes p" and "the user does not believe p" are just different assumptions, which can both either hold or not hold in the user model. But these two assumptions are closely related, because logically they cannot hold at the same time. Hence, each of these assumptions, when holding in the user model, can be regarded as a graduation of the system's beliefs about the user believing p. Then, the third level of degree is that neither of these assumptions hold true. Thus, three-level graduated assumptions SB*UX can be represented using a logical formalism where negation can occur within assumption type identifiers, like in SB∼UB (for "the system believes that it is not the case that the user believes"). SBUX* assumptions can be represented with negation applied to assumption contents ("the user believes p" vs. "the user believes (not p)"), which is only possible if the formalism for assumption contents permits negation.

Furthermore, the use of logical formalisms does not prevent general graduation from being used, where a larger (perhaps infinitely large) number of graduation levels shall be distinguished. Quite a lot of systems use a set of numerical or symbolic values for a fine-grained graduation of user model contents. Representation of these values and, in particular, calculations involving these values go beyond the limits of classical logical formalisms. However, in many cases, the assumptions themselves can be formulated logically, and the graduation values can be regarded as external values that are associated with assumptions. Consider an assumption about the user's belief in a proposition p, i.e. SBUB:p. As described above, such an assumption can be graduated relative to both agent-modality pairs of the assumption type label, SB and UB. Thus, SBUB, SB*UB, SBUB*, and SB*UB* assumptions can be represented, if two places for graduation values are reserved in combination with SBUB contents:

$$\text{SBUB:}p[g_{SB}, g_{UB}]$$

So, all graduation variants of SBUB assumptions are captured with this notation. In general, for an assumption type with n agent-modality pairs, AT^n, and an assumption content ac, the notation is

$$AT^n : ac[g_1, \ldots, g_n]$$

It is certainly possible to implement a mechanism that stores graduation values along with the logically represented contents of one assumption type, in order to accomplish the *representation* of graduation. A more difficult problem is to integrate graduation into logical reasoning processes. In general, the inference engine of a logic-based formalism implements a derivability relation between formulas:

$$A_1, \ldots, A_r \vdash B$$

If the antecedents $A_i, \ldots, A_r$ are graduated, then their graduation values can be used to compute those of the conclusion B. In addition to the logical inference algorithm, an external procedure is required that combines r graduation values to one. Formally, this could be described with

$$A_1[g_1], \ldots, A_r[g_r] \vdash B[f(g_1, \ldots, g_r)]$$

where f is a value combination function. With graduated assumption type expressions, the situation gets slightly more complex, but not fundamentally different (we assume assumption types AT_i^n of a fixed length n; without this assumption, things are somewhat more complicated but work similarly):

$$\begin{aligned} AT_1^n : ac_1[g_{11}, \ldots, g_{1n}], \ldots, AT_r^n : ac_r[g_{r1}, \ldots, g_{rn}] \\ \vdash AT_{r+1}^{m'} : ac'[f(g_{11}, \ldots, g_{r1}), \ldots, f(g_{1n}, \ldots, g_{rn})] \end{aligned}$$

This thesis, however, focuses on logic-based representation; graduation of assumptions will only be dealt with by means of logical negation. The discussion of the previous paragraphs was to show that logical representation and reasoning techniques can be combined with external methods for uncertainty reasoning. This idea is not new; it is in line with the combination of production rules with certainty factors that was introduced in the expert system MYCIN [Shortliffe and Buchanan, 1975]. Similar mechanisms have also been employed in user modeling. [Beaumont, 1994] uses the MYCIN-like production rule language OPS5 [Brownston *et al.*, 1985] for representing rule-like knowledge of medical students in a tutoring system. [Zukerman and McConachy, 1993] use a derivation process to predict the inferences a user can draw from system utterances; in their representation, user beliefs as well as inference rules are graduated, and the strength of a conclusion is computed from the strength of the premises of a rule and the graduation value of the rule itself. The user modeling tool system um [Kay, 1995] only provides quite basic representation mechanisms for user model contents. However, um permits the user model developer to extend the system's capabilities with external tools, especially for the computation of assumption graduations (cf. Section 8.3).

In Chapter 3 we will describe user modeling systems that employ logic-based formalisms for representation. We will also try to characterize systems with extra-logical mechanisms in logical terms. For this purpose, labels like SB*UB will be used for assumption types with graduation, and the above notation for graduated assumption type expressions will be employed.

2.3 The Partition Approach to Belief Modeling

In Section 1.4, the partition approach to belief modeling was already mentioned. Its main characteristic is the distinction of two representation levels. In user modeling, these levels can be used to separate assumption type information from assumption content information in order to allow flexibility especially in the representation of assumption contents. In this section, we will discuss work on user modeling and belief modeling that pursued a partition approach.

2.3.1 Cohen's Nested Contexts

Cohen was the first author to employ a partition approach for modeling the beliefs and goals and nested beliefs and goals of agents [Cohen, 1978]. He uses a separate context for each belief and goal nesting, which contains what is (nestedly) believed or wanted. His representation system is based on [Hendrix, 1975]. Hendrix extended a semantic network formalism for representing belief contents with spaces (i.e., subcontexts) to represent negation, disjunction, and universal quantification.

In order to represent beliefs and goals of system, user, or other agents, Cohen makes use of nested belief and goal contexts. The outer context, which contains content representations and all inner contexts, is the system context SB. Beliefs and goals of other agents, like the user U, are considered to be part of system beliefs. So, for representing beliefs of the system about beliefs of the user, a separate context SBUB is introduced inside the SB context. Other agents than S and U are possible. The assumed beliefs of the user about the beliefs of a third agent A are modeled by means of a context SBUBAB that is placed inside the SBUB context. Besides beliefs, also goals of agents can be modeled. Thus, contexts like SBUW, but also SBSW and SBUBUW are allowed. SBUW and SBSW are inner contexts of SB, and SBUBUW is an inner context of SBUB. The only exception is that there are no "outer goals" of the system, i.e., there is no SW context. Everything, including system goals, is a belief of the system, so that the contents of system goals are contained in the SBSW context.

Like subcontexts for negation and disjunction, inner belief and goal contexts are integrated into the semantic network structure. 'believe' and 'want' nodes are introduced, with an 'agt' link pointing to an agent symbol like U, S or any other agent identifier, and an 'obj' link pointing to an inner context. Figure 2.1 illustrates how most of the above-mentioned belief and goal contexts get organized into an overall representation structure ('agt' and 'obj' link labels are omitted for better readability).

Contexts contain structures of the semantic network formalism, among them 'believe' and 'want' nodes. The 'obj' link of such a node is not restricted to point to inner contexts only. So, *general properties* about the beliefs or goals of agents can be modeled. For example, it is a standard in Cohen's approach that belief contexts (SB...XB) contain a 'believe' node with 'agt' X, and with the 'obj' link reflexively pointing to the context itself. So, the "reflexive" inner context SB...XBXB becomes identical to the containing context. In this way it is represented that an agent, who is modeled to believe in something, is also modeled to believe that she believes it. An expression p contained in context SBUB (i.e., the assumption type expression SBUB:p) then entails SBUBUB:p etc. However, in principle reflexivity nodes can be omitted—they may be contained in a context or not.

The introduction of 'believe' and 'want' nodes as standard elements of the

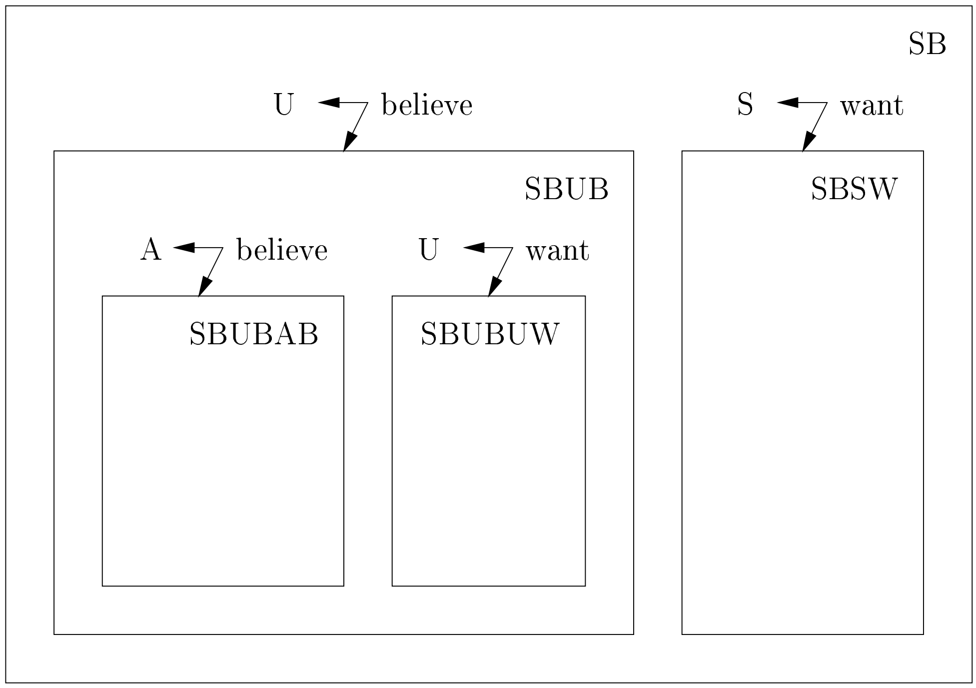


Figure 2.1: A context hierarchy in Cohen's formalism

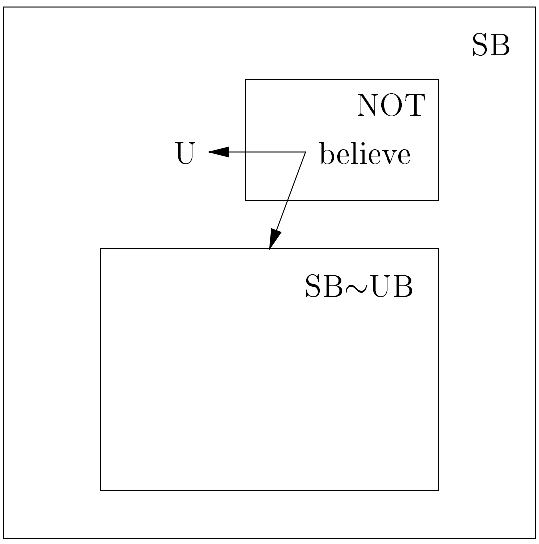


Figure 2.2: Representation of negative assumptions using Cohen's contexts

content formalism allows several other interesting constructs. 'obj' links of believe nodes may not only point to the containing context, but also to outer contexts. This can be employed to represent infinite-reflexive and mutual belief [Kobsa, 1985]. Furthermore, with the help of negation and disjunction subcontexts, beliefs about what an other agent does *not* believe or want and disjunctive assumptions about the beliefs of other agents can be represented. Figure 2.2 illustrates the use of a negation subcontext (NOT) within the SB context for the representation of SB$\sim$UB assumptions. Within the SB context, a NOT context is used. It contains a 'believe' node, with its 'agt' link pointing at U and its 'obj' link pointing to a context SB$\sim$UB. Note that both the agent U and the context for the items that U is assumed not to know are outside the NOT context; only the belief as such is to be negated. Cohen's network formalism is quite powerful; however, the use of contexts especially for disjunction and negation may quickly lead to very complex context structures. Moreover, by the integration of contexts into the semantic network formalism, assumption type and content information get mixed. Nevertheless, Cohen's approach can in principle be regarded as using two levels of representation. It is imaginable that different formalisms are used within contexts. However, the semantic network formalism is mandatory, since it is used to establish context nesting.

2.3.2 Contexts and Acceptance Attitudes in VIE-DPM

VIE-DPM [Kobsa, 1985] was developed as user modeling component of the natural language dialog system VIE-LANG. Like Cohen's system, the belief and goal representation in VIE-DPM is based on the partition approach. I.e., the contents of any belief or goal assumption type, be it a simple or nested type, together form a separate context. However, the contexts of VIE-DPM are not containers of representation elements of a content formalism. In contrast, elements of the current or an intended situation are described outside of contexts using the terminological language KL-ONE [Brachman and Schmolze, 1985]. Contexts contain a number of *acceptance values* that express the attitude of a modeled agent (on some level of nesting) towards the individual objects and the relations between two individual objects in the situation described. Kobsa uses the acceptance values `+`, `-`, and `0`. `+` stands for "accepted", `-` stands for "not accepted", and `0` stands for "uncertain". Each value refers to an object or a relation; it is linked to a representation element that denotes this object or relation (an individual concept or an individual role in terms of KL-ONE). The complete meaning of such a construct is: "The agent accepts that / does not accept that / is uncertain whether (in the situation described) there is an object / a relation between two objects which has all the properties that are defined for it in the situation description." The basic principle of contexts with acceptance attitudes is illustrated in Figure 2.3. The box represents a context, which contains four acceptance values. The values refer to representation elements of KL-ONE (indicated by the ovals in the upper part

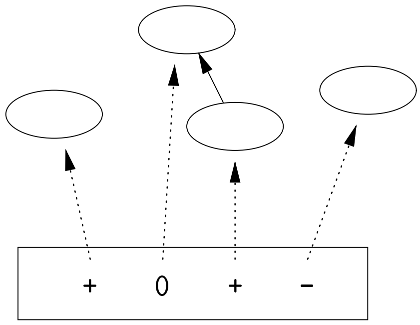


Figure 2.3: A simple context in VIE-DPM

of the figure).

Assumptions about beliefs and goals of other agents are also represented with the help of contexts. Every context has a name, like SB, SBUW, etc., and is marked with its general "assumption class", i.e. 'B' for a belief context and 'W' for a goal (want) context. A distinct acceptance value is linked to the representation element that denotes the agent of the context, i.e. the user, the system, or another agent. SB is the outer context of a VIE-DPM representation. It is linked to inner contexts like SBUB or SBUW. The acceptance values of inner contexts are composite values according to the level of nesting. For instance, a value in SBUB may be +-, the meaning of which is that the system believes that the user does not believe that there is an object or a relation with the described properties. For a context on the third level of nesting, like SBUBSB, acceptance values are composed of three single values, and so on. Note that through the use of acceptance values, the contexts of VIE-DPM correspond to assumption types with graduations at any level of nesting, like SB*, SB*UB*, SB*UW*, SB*UB*SB*, etc.

Similar to Cohen's approach, VIE-DPM represents mutual belief by links between contexts that point backward in the sense of the belief nesting. So, if an "inner context" link of SBUBSB points to a belief context of the user, then this will normally be SBUBSBUB. If the link points back to SBUB instead, this means that SBUB and SBUBSBUB become identical. Following the link from SBUB (= SBUBSBUB) to the next belief context of the system, we reach SBUBSB, which is therefore identical with SBUBSBUBSB, and so on.

Another important representational element of contexts are links between acceptance values, which can point from a context into an inner context or into another context at the same level of nesting. These links are to express that the objects, which the acceptance values refer to, are related to each other. These *inter-nexus links*[1] can be used for special representation purposes that we will not discuss in detail. For example, from one value in SB, two inter-nexus links

[1] The structure that carries an acceptance value is called *nexus*.

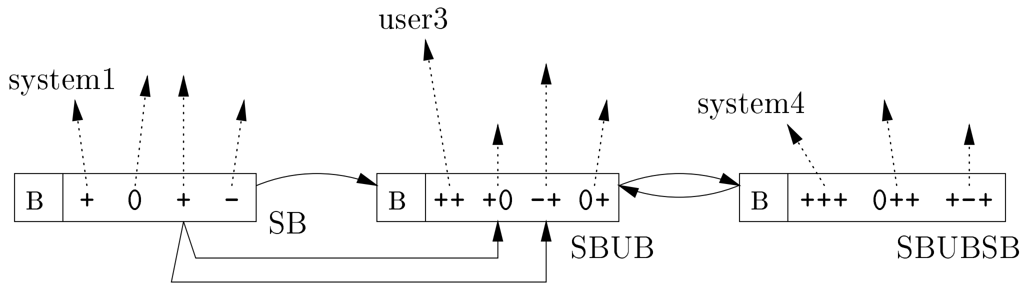


Figure 2.4: Representation elements of VIE-DPM

may point to two values in SBUB. This can be interpreted such that the system knows that the two objects, which the values in SBUB refer to, are identical, but that it believes that the user believes them to be different.

Figure 2.4 summarizes the representation facilities of VIE-DPM that have been described above. There are three contexts, SB, SBUB, and SBUBSB. SB is linked to SBUB, SBUB is linked to SBUBSB, which is linked backward to SBUB. The interpretation of acceptance values is as described. There is only one pair of inter-nexus links that illustrates the situation mentioned above.

2.3.3 Partition hierarchies in BGP-MS

In early work on the user modeling shell system BGP-MS, Kobsa introduced partition hierarchies as means for user model representation [Kobsa, 1990]. In contrast to the contexts of VIE-DPM, a partition does not refer to knowledge representation elements but *contains* such elements. The partition mechanism SB-PART [Scherer, 1990], which was employed in the early days of BGP-MS, was specialized to allow structures of the KL-ONE-like language SB-ONE [Kobsa, 1991] as partition contents. In contrast to the above-mentioned approaches, BGP-MS partitions do not have nested inner partitions. Instead, subordination relations between partitions can be represented by inheritance links; i.e., partitions can inherit the contents of other partitions. With inheritance links, partitions can be arranged in arbitrary hierarchies like those shown in Figure 2.5, which will soon be explained in more detail.

With partition hierarchies, it becomes possible to store the common contents of two or more partitions in a common superpartition, from which these contents can be *inherited*. According to Kobsa, this minimizes redundancy in the knowledge representation. Avoidance of redundancy is supported by the *propagation* mechanism: Contents that are common to all subpartitions of one partition P are propagated into P, from which they will be inherited back to the subpartitions. Taking inheritance into account, partitions are not isolated; accessing the contents of a partition P always means accessing the contents of P plus those of all of its superpartitions. Kobsa assigns a special role to the leafs of partition

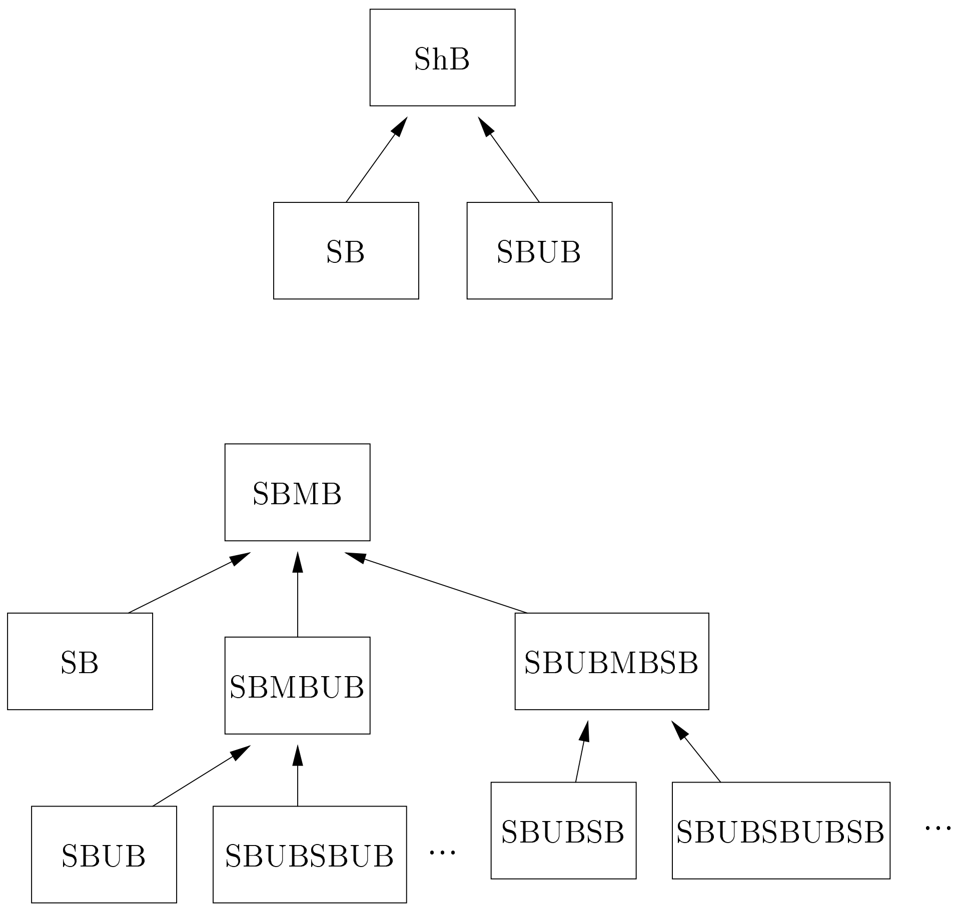


Figure 2.5: Partition hierarchies in BGP-MS

hierarchies: All partitions that can be reached from a leaf partition following the inheritance links (including the leaf partition itself) form a so-called *view*. Views are important because of a specific property of SB-ONE: Its elementary constructs need to be combined with other constructs to form syntactically well-formed SB-ONE expressions. Particularly through propagation, single SB-ONE constructs can become moved across the partitions of a view, so that in single partitions syntactically ill-formed expressions may be present. Therefore, Kobsa requires syntactical well-formedness only for the contents of a view, i.e. for the union of all structures in the partitions of a view.

If employed for user modeling, partitions can carry the usual labels SB, SBUB, etc.[2] Kobsa demonstrates the utility of partition hierarchies for user modeling. He illustrates, how they can be employed for representing shared beliefs (with a 'ShB' partition) of system and user, expert knowledge of the system (SB), and misconceptions of the user (SBUB). Furthermore, complex partition hierarchies can be used for representing mutual beliefs of system and user. Similar to Cohen, Kobsa regards all user model contents as beliefs of the system, so that the main mutual belief partition is called SBMB (the system believes that it is mutually believed). Mutual belief assumptions are defined as the union of all assumptions of type SB, SBUB, SBUBSB, etc. Sub-categories of mutual belief are SBMBUB and SBUBMBSB; SBMBUB is the intersection of all {SBUB}* beliefs, and SBUBMBSB is superordinate to all SB{UBSB}* beliefs. So, SB, SBMBUB, and SBUBMBSB can be the direct subpartitions of SB in a hierarchy with mutual beliefs (cf. Section 2.1). For practical applications, not all of these partitions need to be present. The developer of a user modeling system can define arbitrary partition hierarchies with BGP-MS (as documented in [Kobsa, 1990]), he may even use arbitrary partition names. This means that no theory of belief representation is pre-implemented into BGP-MS. Figure 2.5 shows both the simple hierarchy discussed above and a hierarchy for mutual beliefs.

A further use of partitions in BGP-MS is the representation of stereotypes. Basically, stereotypes [Rich, 1983] are containers of assumptions about potential user subgroups (cf. Section 4.3 for a detailed discussion of stereotypes). If a stereotype applies to an individual user, it becomes *activated* and its contents must be integrated into the individual user model. It is straightforward to use partition hierarchies for stereotyped assumptions. A partition can contain the assumptions that belong to one stereotype, and the integration into the individual model can be done by introducing an inheritance link from the stereotype partition to a partition of the individual model. Furthermore, stereotype hierarchies can be represented by a hierarchy of stereotype partitions. In [Kobsa, 1990], it is not clearly said, how a stereotype partition is exactly linked to the partitions of the individual model. Kobsa only mentions belief partitions; so, a stereotype

[2]In [Kobsa, 1990], goal partitions like SBUW are not mentioned. However, there is no evident obstacle to the use of such partitions in the partition mechanism as presented in that paper.

partition will probably contain basic assumptions about the beliefs of the members of a certain group. Then, in case of stereotype activation, the stereotype partition will inherit to the SBUB partition.

2.3.4 Ballim's Nested Belief Models

In several publications, Ballim and Wilks deal with the ascription of belief in nested belief models involving a number of agents [Ballim and Wilks, 1991b; Ballim and Wilks, 1991a; Ballim, 1992]. A belief model is a kind of context that is associated with an agent. It contains knowledge items that are believed by the agent. A context may also contain inner contexts, that are associated with other agents. Inner contexts then contain the beliefs that the outer agent ascribes to the inner agent. Contexts may be nested to an arbitrary depth, although Ballim states that he does not expect nestings with a depth greater than six or seven [Ballim and Wilks, 1991b, p.51]. A nested model is denoted by a sequence of agents, $< \alpha_1, \alpha_2, \ldots, \alpha_n >$. Since Ballim and Wilks deal with belief models only, the modality need not be mentioned in identifying a nested model. Moreover, since all models are regarded to be held by the system, the system needs not be listed as first agent.

Belief ascription means that an agent forms an idea (a belief) about what another agent believes. [Ballim and Wilks, 1991b] state that an obvious way for ascribing beliefs to other agents is to use their utterances or actions as basis. However, the interpretation of utterances and actions may already depend on what is believed about the other agent. So, it becomes necessary to generate hypotheses about other agents that are not safely based on observations, particularly at early stages of interaction. Ballim and Wilks discuss "two predominant methods" for dynamically forming such hypotheses.

First, stereotype models about groups of agents can be applied. The contents of a stereotype model will be ascribed to an agent, if the agent is determined to be a member of the corresponding group. However, when nested models are considered, belief ascription with stereotypes may become quite expensive. The reason is that the system must reason about the ability of inner agents to apply stereotypes. That is, the stereotypes of inner agents may differ from those of the system, and an inner agent may ascribe stereotypes to other agents than the system would do. In order to correctly generate hypotheses about nested belief models, stereotype information would have to be transformed while stepping down the levels of nesting. It is an interesting insight that in models involving several agents, stereotypes may be relative to the individual ascribing agent. [Hustadt, 1995] develops a modal logic extension for a description logic, where agents and agent groups are modeled as individuals and concepts of the logic; this allows one to flexibly express individual stereotype contents as well as individual stereotype ascription conditions (for details, see Section 3.4.2).

Second, an agent may simply ascribe its own beliefs to other agents, possibly in a perturbed form. Nested models are generated based on the assumption that other agents do the same. This method is called "perturbation". It is a kind of default reasoning; the standard ascription rule is to ascribe one's own beliefs to another agent except if there is explicit evidence that the other agent is already assumed to hold a contradicting belief. This rule is quite straightforward; however, it is not valid for beliefs that the ascriber considers to be atypical. These are beliefs that should only be ascribed to another agent when there is evidence to do so. For example, expert knowledge, especially in uncommon domains, should not be ascribed to other agents without a reason. It is interesting to note that stereotype models often cover specialized, atypical beliefs that can be ascribed to specific groups of agents only. So, the perturbation method could be improved by mixing it with stereotype ascription. However, Ballim and Wilks show that the complexity of the perturbation method increases dramatically when stereotypes are introduced. The reason is that in ascribing beliefs to an inner agent, the stereotypes of all outer agents in the nesting hierarchy must be considered, as well as the stereotype models that stereotypes (i.e., all members of a stereotype) may hold.

[Ballim and Wilks, 1991b] describe a method for belief ascription that amalgamates the stereotype and perturbation approaches. It is based on a notion of competency; i.e., an agent will be believed to hold a belief if it is assumed to have the competency to believe it. Ballim and Wilks employ lambda expressions in combination with so-called evaluation relations in order to express that some agent is competent to evaluate some expression to some value and therefore can be assumed to believe that the expression has the value. A simple example is (taken from [Ballim and Wilks, 1991b]):

$$(\lambda.x1(\text{Lisp-progr } x1) \text{ John}) \ [(\text{System,John,Mark}),\text{TRUE}]$$

consists of the lambda expression $(\lambda.x1(\text{Lisp-progr } x1)\text{ John})$ and of an evaluation relation between three agents and one value, [(System,John,Mark),TRUE]. The combined expression means that the agents System, Mark, and John are competent to evaluate the expression '(Lisp-progr $x1$)', with $x1$ bound to John, to TRUE. That is, they believe that '(Lisp-progr John)' holds.

It is possible that agents evaluate expressions differently. Moreover, stereotypes can be mentioned in evaluation expressions:

$$\begin{array}{l}(\lambda.x2(\text{useful } x2) \text{ Lisp}) \\ \quad \text{comp}\left(\begin{array}{l}[(\text{System,Lisp-progr}),\text{TRUE}] \\ [(\text{C-progr}),\text{FALSE}]\end{array}\right)\end{array}$$

Here, the complex evaluation relation 'comp(X,Y)' is employed, which means that there are competing evaluations. Agent symbols may denote groups of agents, like 'Lisp-progr' and 'C-progr'. So, the above expression means that the agents 'System' and 'Lisp-progr' believe that Lisp is useful, and that 'C-progr' believes that this is not the case. In the process of ascribing beliefs to John, the system would check the evaluation relations of all available expressions. In this example, the result is that John believes himself to be a Lisp programmer, and that John believes that Lisp is useful (because the system also believes that John is a Lisp programmer, and Lisp programmers believe Lisp to be useful). More difficult evaluation relations can be formed. In some cases, evaluation relations need to be transformed during the process of belief ascription, e.g. when the evaluation relations contain information that the modeled agent is not aware of. For more details, see [Ballim and Wilks, 1991a].

Note that with the use of evaluation relations, belief models of both individual agents and agent groups are no longer collections of belief items like in the work of Cohen and Kobsa. The information about "who believes what" is not maintained by having a partition-like container of items; here, it is distributed by attaching evaluation relations to single items. Nevertheless, this work can be regarded as pertaining to the partition approach. The common characteristic of all partition-based systems is that there are two levels of representation. On the one level, there is information about the agents and the modalities that are considered in a model: In [Ballim and Wilks, 1991b], only beliefs are considered, but arbitrary agent symbols are allowed. Cohen and Kobsa permit modalities B and W for beliefs and wants, respectively. In the work of both authors, system and user are "special agents" that play a dedicated role in the context of user modeling; in addition, Cohen permits the modeling of third agents. On the other level, information is expressed about what is believed or wanted: Ballim and Wilks use lambda expressions, Cohen employs semantic networks, and Kobsa employs a KL-ONE-like representation. Two-level representations for agent modeling fit our distinction between assumption type and assumption content as the two central components of an assumption about the user (see Section 1.3).

2.4 Combining Partitions and Modal Logic

Partitions or Modal Logic?

As the discussion in the previous section shows, partition-based representation systems seem to be quite appropriate to the task of user modeling. However, partition-based systems have been criticized mainly because of their deficiencies concerning expressive power and reasoning abilities. [Kobsa, 1992] admits such deficiencies for the user modeling shell system BGP-MS, which was discussed in the previous section. On the one hand, the partition hierarchies of BGP-MS implement a number of frequently-used inferences like inheritance and propagation. On the other hand, it is difficult, if not impossible, to express negative assumptions (the user does not hold a belief), disjunctive assumptions (the user believes p_1 or believes p_2) and general implications (if the user believes p_1, then she also believes p_2).

The partition approach is by far not the only method for modeling beliefs and other attitudes of agents. On the contrary, logic-based approaches are much more prominent in the area of reasoning about knowledge and beliefs. In Section 1.4, the modal logic approach to belief modeling was mentioned. Since the introduction of epistemic logic by Hintikka [Hintikka, 1962], many variants of modal logic and other logical systems have been employed for modeling beliefs (for an overview, see [McArthur, 1988; Reichgelt, 1989; Fagin *et al.*, 1995]). [Kobsa, 1985] demonstrates shortcomings of several logical belief representation systems in comparison with the partition-based system VIE-DPM. However, modal logics or epistemic logics can provide all the facilities that were mentioned above as not possible in a partition-based system. Let B_X be a modal operator meaning "agent X believes", and let us assume a logical language with the standard connectives[3]. Then, a simple modal expression like $B_S B_U\, p$ ("the system believes that the user believes p") can be regarded equivalent to an assumption type expression SBUB:p. However, modal logic allows more complex expressions:

$$B_S \neg B_U\, p$$

means "the system believes the user does not believe p",

$$B_S B_U\, p_1 \vee B_S B_U\, p_2$$

stands for "the user is assumed to believe p_1 or assumed to believe p_2", and

$$B_S B_U\, p_1 \rightarrow B_S B_U\, p_2$$

represents the rule "if the user is assumed to believe p_1, then she can also be assumed to believe p_2". Moreover, by using further operators like W_X for 'agent X wants', other attitudes of the user can be modeled. Operators can be nested, like B_S and B_U above, to express belief nestings, and since operators can occur everywhere inside of complex formulas, the expressive possibilities are almost unlimited. The perhaps main advantage of modal logic, however, was only indicated by the above examples: Representation and reasoning is not confined to the belief space of a single assumption type, i.e. to a single partition or, in the partition inheritance hierarchies of BGP-MS, a single view. In the above example, two "type-internal" expressions of the same type, namely system assumptions about user beliefs (SBUB), were connected, which already goes beyond representation within one partition (e.g., $B_S B_U p_1 \vee B_S B_U p_2$ is different from $B_S B_U(p_1 \vee p_2)$). But also different assumption types can be combined, if desired; e.g.,

$$B_S W_U\, p_1 \rightarrow B_S B_U\, p_2$$

expresses that an assumed user goal p_1 implies that the system can assume the user belief p_2.

[3] A precise definition of a modal language will be given in Section 3.4.

So, modal logic is a powerful formalism, but also a very general one. Reasoning with modal logic is a hard task and, in case of a modal predicate logic, an even undecidable problem. One motivation for the use of partitions is that user model representation and reasoning tasks often only involve one type of assumptions, and can therefore be handled within one type. The assumption contents of one type can be represented with a formalism that, unlike modal logic, needs not express information about the type of an assumption. In both the VIE-DPM and the BGP-MS system, Kobsa employs a KL-ONE language for representation of concept-based partition contents, the utility of which he demonstrates with several examples. KL-ONE languages usually provide several built-in inferences that are mainly based on the notion of concept inheritance. However, terminological languages like KL-ONE also have their limitations; negation, disjunction, and quantification cannot be expressed in full generality. So, there is a conflict between using an expressive and powerful, but very general formalism like modal logic, and using specialized and efficient mechanisms like partitions and KL-ONE, which unfortunately have deficiencies as far as expressiveness is concerned.

Top-Down Integration of Partitions and Modal Logic

For BGP-MS, [Kobsa, 1992] suggests a hybrid representation approach that on the one hand is a remedy of the shortcomings of both partition hierarchies and the content formalism SB-ONE, but on the other hand preserves the specialized and efficient representation and reasoning facilities of both subsystems. Kobsa proposes to use modal logic as the principal representation, in the form of first-order predicate calculus expressions that are translations of modal formulas. Such a translation expresses the properties of the standard possible worlds semantics of modal logic in first-order logic. An early translation technique is the *relational translation* [Moore, 1980] that represents the accessibility relation of the possible-worlds semantics with a designated first-order predicate R. For BGP-MS, *functional translation* [Ohlbach, 1991] was to be used instead, which represents the accessibility relation with first-order functions. According to Kobsa, the main advantage of this technique is that, for a translated formula, it can easily be detected whether the formula can be represented as content of one partition. Thus, e.g., the modal formula $B_S B_U p$ can be associated with partition SBUB, and it can be represented by storing p into SBUB. Now, in a modal predicate logic, p can be a complex first-order formula. So, first-order predicate calculus (FOPC) is needed as formalism for partition contents. By using FOPC within partitions, the shortcomings of SB-ONE concerning expressive power are overcome. But SB-ONE is preserved: if an FOPC partition content is representable in SB-ONE, it is translated into SB-ONE structures. In summary, modal formulas are reduced in a top-down manner as far as possible, first to FOPC partition contents, and then to SB-ONE partition contents. With this *top-down* approach to synthesizing modal logic with partitions and SB-ONE, expressive power is available, but the benefits of the specialized inferences of both partition mechanism and SB-ONE are preserved.

The top-down approach to integrating partitions and modal logic was pursued in the further development of BGP-MS. Instead of the functional translation of modal formulas, expressions of the "Belief and Goal Description Language" (BGDL), a special modal logic language, became the principal means for describing user model contents in entries and queries to a BGP-MS user modeling knowledge base [Kobsa and Pohl, 1995]. As the name of this language suggests, two modal operators B for beliefs and W for goals/wants can be used to formulate UMKB contents. Internally, partition hierarchies together with FOPC and SB-ONE as content formalisms are available. Following the ideas of [Kobsa, 1992], BGP-MS always attempts to handle BGDL expressions with the most specific mechanisms possible. I.e., BGP-MS prefers SB-ONE partition contents over FOPC partition contents over modal logic formulas as means for the internal representation of BGDL expressions. As far as reasoning is concerned, reasoning with partition contents is the standard mechanism of BGP-MS. As alternative, however, modal logic reasoning can be employed. For this purpose, the functional translation technique was automatized and made applicable to the whole range of BGDL expressions.

Bottom-Up Combination of Partitions and Modal Logic

This thesis makes use of several of the ideas that are described in [Kobsa, 1992], and it continues work on BGP-MS as described in [Kobsa and Pohl, 1995]. In contrast to previous work, however, it will mainly pursue a *bottom-up* approach of extending partition representation with modal logic capabilities, strictly founded on a clear formal framework. Partial knowledge bases, which are called *assumption types*, are the basic building blocks of this *Assumption Type Representation* (AsTRa) framework for user model representation, which is introduced in this thesis. Strictly speaking, assumption types are not different from partitions, but the name shall explicitly suggest that one container stores exactly all assumption contents of one type that are present in the user modeling knowledge base. In principle, assumption contents can be represented with an arbitrary number of content formalisms. These formalisms are demanded to be logic-based, i.e. to provide knowledge base access and inference functions and to be characterizable in logical terms.

The range of possible assumption types will be clearly defined. In contrast to the partitions of previous partition-based systems, assumption types will be independent of each other. These properties make it possible to characterize AsTRa knowledge base contents in terms of modal logic formulas. Thus, AsTRa becomes the first partition-like approach with a clear formal semantics, which is one of the main contributions of this work. Furthermore, based on the correspondence between AsTRa and modal logic, modal reasoning techniques can be applied to process (the modal logic correspondences of) assumption type expressions together with more complex modal formulas. We will also utilize functional translation for modal reasoning and will exploit the relationship between translated expressions and assumption types that Kobsa similarly observed for partitions.

The AsTRa framework combines partition-based and modal logic-based representation in a bottom-up way. Bottom-up means that normally specialized facilities, i.e. assumption types and assumption content formalisms, are accessed and used explicitly instead of implicitly accessing them via modal logic formulas. However, modal logic representation and reasoning, which are the more general and powerful mechanisms, can be employed if necessary or desired.

AsTRa is designed as a representation framework for user modeling shell systems. Shell systems should not only provide expressive power for the representation of user models. In addition, they ought to offer flexibility in the sense that a user model developer is allowed to choose from a range of representation and reasoning alternatives those that best fit the needs of an application system. The bottom-up approach and the clear logical foundation of AsTRa offer this flexibility:

1. It will not let modal formulas trickle down into partitions if this is possible, but offers a choice between the use of assumption types with assumption contents only and the use of full modal logic capabilities.

2. It will not reduce content expressions of one formalism to content expressions of a more special formalism if possible, but permits to explicitly use several formalisms within assumption types in parallel.

3. For modal reasoning, it employs a version of functional translation that can process arbitrary modalities. To this respect, it exceeds the suggestions of [Kobsa, 1992], where translated modal expressions allowed the representation of one modality (belief, goal, etc.) only. Hence, also assumption types are not restricted to beliefs or goals only, as in the BGDL language of [Kobsa and Pohl, 1995]. Modalities can freely be added to an AsTRa system. As a side effect, AsTRa allows stereotypes to contain assumptions not only about beliefs, like in BGP-MS [Kobsa, 1990; Kobsa and Pohl, 1995], but also about goals, preferences, etc.

4. Experience with user modeling applications has shown that in many cases, the only representational need that goes beyond standard partition mechanisms is the representation of negative assumptions. Based on the modal logic semantics of assumption types, additional negative assumption types are introduced along with corresponding specialized reasoning mechanisms. Negative assumption types can be used instead of or in combination with modal reasoning techniques. Negative assumption types are different from standard partitions as far as reasoning with assumption contents of one

type is concerned. So, special reasoning mechanisms are developed for negative types based on their modal logic semantics. Together with these negative assumption type reasoning mechanisms and full modal reasoning, the AsTRa framework provides five levels of reasoning from which a user model developer can freely choose, according to the specific needs of his application.

5. The semantics of a specific modal logic is traditionally defined via a set of axioms. In the area of automated deduction systems, techniques for mechanizing the introduction of axioms into a logic have been developed. [Allgayer *et al.*, 1992] propose to exploit such techniques in order to be able to freely use quite arbitrary logics for a user or agent model. This thesis follows that proposal. It employs a procedure for translating modal logic axioms into first-order logic such that they can be processed together with translated assumption type expressions and modal formulas. So, a user model developer is able to add modal axioms to the user modeling knowledge base, thus defining the meaning of the used modalities as well as general relationships between assumption types.

6. Finally, the top-down approach of [Kobsa, 1992] is not abolished but retained as alternative. The UMKB access functions of AsTRa will be able to process modal formulas, so that the user model developer can let the system determine the assumption type and appropriate content formalism for a given assumption about the user.

The critique of partition-based approaches is partly summarized in [Hustadt, 1995], who basically mentions the following three items:

- Partitioned knowledge bases have no formal semantics.
- All reasoning is confined to one partition.
- Partition inheritance is implemented by an ad hoc mechanism which cannot be controlled by the knowledge engineer.

The AsTRa framework tackles all these issues. It provides a semantics for the basic assumption type representation. It extends and combines contents of assumption types with complex modal formulas to enable reasoning beyond assumption types.

The AsTRa framework has been implemented in the most recent version of the user modeling shell BGP-MS (see Chapter 6). In BGP-MS, assumption types are represented using a partition mechanism with inheritance and propagation procedures. Thus, BGP-MS implements inheritance and propagation inferences between assumption types. The modal logic semantics of AsTRa allows the formal characterization of both inheritance and propagation relations between assumption types. As in earlier versions of BGP-MS, SB-ONE and FOPC are employed as content formalisms. However, an integration of further content formalisms is possible following the specifications of the AsTRa framework.

The details of the AsTRa framework will be specified in Chapter 5. Before, logic-based representation and reasoning methods for user modeling will be investigated in form of a review of the relevant user modeling literature. In particular, representation methods will be related to the assumption type representation approach in order to demonstrate the applicability of the AsTRa framework.

Chapter 3

Logic-Based Knowledge Representation for User Modeling

This chapter presents user modeling systems that have been described in the literature to make use of logical formalisms for the representation of assumptions about the user and of other user modeling knowledge . The formalisms that are mentioned are ordered according to their expressional power. The discussion starts with simple propositional logic and gradually steps forward to modal logics, particularly considering epistemic logic.

At the same time, this chapter aims at showing the potential of logic-based formalisms for user modeling. It presents how logic-based representation techniques have been and can be employed in user modeling systems. Some of the systems discussed make use of quite specific representations. For these systems, it will be demonstrated that their representation facilities are related to logic-based formalisms. Furthermore, we will attempt to characterize the user modeling knowledge base contents of most systems in terms of assumption types and assumption contents. It will turn out that almost all systems could make use of an assumption type representation. Some systems would additionally need extensions for representing inferential relationships between assumption types.

For each logical formalism considered, there will be a formal introduction with a definition of its syntax and semantics. Afterwards, examples of its use in user modeling systems will be presented. The basis of this presentation is an analysis of the user modeling literature; all papers from [Kobsa and Wahlster, 1989], which presents the most important collection of early user modeling work, and from the journal ‘User Modeling and User-Adapted Interaction’ were considered, as well as several relevant papers from the proceedings of the international workshops and conferences on user modeling and of other conferences.

3.1 Propositional Calculus

3.1.1 Introduction

A proposition is a sentence that can be said to be true or false, e.g. "snow is black" or "5 is a prime number" [Hilbert and Ackermann, 1972]. Propositional calculus treats such sentences as basic entities. Propositions that are built from other, more simple propositions, like "5 is a prime number or snow is black" can be constructed with the help of logical connectives. The language of propositional calculus, as it will be used hereafter, is defined by determining the set of its formulas:

Definition 3.1 (Formulas of propositional calculus) *Any symbol will be allowed as* atomic formula *(short: atom). If A and B are formulas, then also*

1. (A)
2. $\neg A$ *(not A, negation)*
3. $A \wedge B$ *(A and B, conjunction)*
4. $A \vee B$ *(A or B, disjunction)*
5. $A \rightarrow B$ *(if A then B, implication)*
6. $A \leftrightarrow B$ *(A if and only if B, co-implication/equivalence)*

are formulas. 1 – 6 are the only ways to form formulas of propositional calculus.

In the following, a strict binding precedence within the set of connectives is assumed, following the order of connectives in the above definition. E.g., $A \vee B \rightarrow C$ is short for $(A \vee B) \rightarrow C$, but not for $A \vee (B \rightarrow C)$. Thus, parentheses need to be used only for building formulas that deviate from standard precedence.

The semantics of a logical calculus deals with the truth of formulas. So, mappings of formulas to truth values are the central semantic instrument; such mappings are called interpretations. Interpretations only differ in their assignment of truth values to atomic formulas. For the syntactical operators of a calculus (in propositional calculus: the connectives), it is defined how truth values of complex formulas that were built using the operators are determined from the truth values of their sub-formulas.

Formulas of propositional calculus are interpreted as follows:

Definition 3.2 (Interpretation) *In propositional calculus, an interpretation $\mathcal{I}$ is a total function I that maps all formulas to a truth value. Basically, it assigns a truth value to every atom: $I(A) \in \{true, false\}$ for every atom A. The truth values of complex formulas can be computed from their sub-formulas as follows:*

- $I((A))$ = *true iff* $I(A)$ = *true*
- $I(\neg A)$ = *true iff* $I(A)$ = *false*
- $I(A \wedge B)$ = *true iff* $I(A)$ = *true and* $I(B)$ = *true*
- $I(A \vee B)$ = *false iff* $I(A)$ = *false and* $I(B)$ = *false*
- $I(A \rightarrow B)$ = *false iff* $I(A)$ = *true and* $I(B)$ = *false*
- $I(A \leftrightarrow B)$ = *true iff* $I(A) = I(B)$

A is said to be true in $\mathcal{I}$ iff $I(A)$ = true.

Based on the notion of an interpretation, we will define further concepts that will often be used in the following chapters. The following definitions refer to the abstract notion of an interpretation only, so that they can be transferred to both predicate calculus (Section 3.2) and to modal logic (Section 3.4).

Definition 3.3 (satisfiable, model, entailment)

1. *A formula ϕ is* satisfiable, *iff there is an interpretation $\mathcal{I}$ such that ϕ is true in $\mathcal{I}$. In this case, it is said that $\mathcal{I}$* satisfies *ϕ.*
2. *$\mathcal{I}$ is called a* model *of ϕ, iff $\mathcal{I}$ satisfies ϕ. $\mathcal{I}$ is a model of a set of formulas KB iff it is a model of all formulas in KB.*
3. *A set of formulas KB* entails *a formula ϕ, $KB \models \phi$, iff every model of KB also is a model of ϕ.*

Perhaps the most important connective is the implication. From Definition 3.2 above we can see that if the formulas A and $A \rightarrow B$ are true then also B must be true. So, if a deductive knowledge base (e.g., a user modeling knowledge base) contains the formulas A and $A \rightarrow B$ (which means that they are considered true), from a purely logical point of view also B is implicitly considered true and can be *inferred* from the database. Thus, implications are often called *inference rules*. In user modeling systems, they are often used to make statements about what user model contents can be inferred from current contents.

3.1.2 Propositional Calculus for User Modeling: Atomic Propositions

A considerable number of user modeling systems uses atomic propositions in the formulation of assumption contents, particularly for assumptions about the user's knowledge in a given domain. However, most of them combine propositions with a numerical or symbolic value in order to graduate the user model content. This kind of user model representation has been called "linear parameters" in the literature [Wahlster and Kobsa, 1989; McTear, 1993]. In the following, several examples for the use of atomic propositions in user models are given.

In [Paris, 1989], the user model contains assumptions about the knowledge of a user about objects, i.e. about their functionality and other properties, and his knowledge of basic concepts of the domain. This is represented by having two lists of symbols, one for the objects and one for the concepts the user knows about. The type of assumptions used here is SBUB, and the assumption content can be represented by atomic propositions. Thus, using the suggested notation for assumption type expressions, the symbol 'microphone' being contained in the list of known objects can be rewritten as SBUB:microphone[1].

[Shifroni and Shanon, 1992] represent the user's knowledge of landmarks in a geographic domain with a function KL_U (KL standing for "knows landmark") that maps a given set of landmarks to the set {knows, doesn't know}. Obviously, this can be expressed using assumption types SBUB and SB~UB with atomic propositions for assumption contents.

Several authors use SBUB* assumptions to express how much the user is familiar with concepts, instruments or other objects of the underlying domain. In his intelligent tutoring system (ITS), Nwana uses three graduation levels denoted by the numbers 0 (not known), 1 (fairly known), and 2 (well known) [Nwana, 1991]. For internal representation of student models, his system utilizes a vector with a slot for every concept of the domain; the vector slots can be assigned one of the graduation values. In the discussion of his work, Nwana introduces an external, more symbolic notation: He says his student model to contain expressions of the form sm(*concept,level_of_understanding*) for any concept the vector value of which is 1 or 2, these numbers being the *level_of_understanding* value. An additional relation sm_1(*concept*) ist used to express that the student knows a concept and knows how to apply it. With the use of sm_1, a fourth graduation level is introduced. Concepts, being the assumption contents in this representation, are represented by symbols, which can be regarded as atomic propositions. A user model content sm(*concept*,*n*) can be rewritten as graduated assumption type expression: SBUB*:*concept* [n]

For the adaptive generation of help texts (responses to user queries and explanations), [Tattersall, 1992] assesses the strength of user knowledge about system concepts as 'unknown', 'introduced', and 'known'. Also in this system, atomic propositions are sufficient to represent the contents of SBUB*-type assumptions.

TECHDOC-I is a plan-based help and support system for car maintenance [Peter and Rösner, 1994]. TECHDOC-I employs an object-oriented representation scheme with attribute-value pairs for all its representation purposes. E.g., an object representing a step of a maintenance plan has slots for a name, for a degree

[1] We will further-on use a sans serif font for (partially) concrete examples of assumption type expressions.

of familiarity, for an execution counter, for relations to other plan steps, and for relations to technical objects that are necessary to execute the plan step. User model, dialog history, and system knowledge are merged in such a representation. For representing the user model part, the name of an object and its degree of familiarity, which may be 1 (novice), 2, or 3 (expert), are sufficient. Thus, also the user model of TECHDOC-I can be represented with atomic propositions (the object names) associated with a three-level graduation. Hence, the only assumption type is SBUB*.

Other user modeling systems have used graduated assumptions of type SB*UB in order to express the strength of system belief in its assumptions about user beliefs. The system UMFE [Sleeman, 1985] presents an approach to user modeling that may be generalized and used in other systems. In its user model, all elements of a concept hierarchy (which is part of system domain knowledge) are assigned one of the values 'known', 'not-known', and 'no-information'. So, there are values for positive, neutral and negative evidence about the user knowing a concept. Internally, UMFE uses numerical graduation values ranging from -100 to 100, but actually uses only the values -100 (not-known), 0 (no-information) and 100 (known). Note that this kind of graduation corresponds to what can be expressed with logical negation; UMFE's user model can be represented by using atomic propositions (denoting the concepts) with assumption types SBUB and SB∼UB. Given a concept C, 'known' and 'not-known' would be replaced by having $\mathsf{SBUB}{:}C$ and $\mathsf{SB{\sim}UB}{:}C$, respectively, in the user model. 'no-information' would, quite naturally, correspond to neither of these assumptions being contained in the user model.

Grundy [Rich, 1979; Rich, 1983; Rich, 1989] is one of the first user modeling systems. In Grundy, user preferences concerning book contents are modeled. A user model consists of a set of attributes (also called facets) with user-specific values. There are two kind of attributes: Attributes of the first kind convey general information about the user (e.g., motivations); attributes of this type can have different symbolic values. The other attributes correspond to possible properties of book contents (e.g., thrill); they get assigned a numerical value ranging from -5 to 5. For these "scalar facets", this value expresses the preference strength and leads to an SBUPref* graduation. In parallel, the values of all facets are accompanied by certainty values, which range from 0 to 1000 and represent the strength of the assessment. Altogether, the scalar facets of a Grundy user model can be represented with atomic propositions for denoting the several personal traits; these would be contents of assumption type SB*UPref*. The other attribute-value pairs can be regarded as SB*UProp (meaning: the system believes with a certain strength that the user has some property) assumptions. However, for representing the assumption content in logic, propositional calculus is not enough; logical means that could be used to represent non-scalar Grundy facets are discussed in the next section.

It is interesting to note that for SBUB* assumptions that can be expressed with atomic propositions, positive graduation values are typically used. This is related to the observation that in most systems knowledge about domain concepts is modeled. Such concepts do not have a propositional content of their own and cannot be negated like a real proposition. So, along with domain concepts, only positive graduation values can be used in SBUB* assumptions. However, a concept C being contained in a user model corresponds to the proposition "the user knows C". Graduating this user model content would lead to an SB*UB assumption. Since propositions like the one above can be negated, along with domain concepts negative graduation values are possible in SB*UB assumptions. In contrast, Grundy uses negative graduations of SBUPref* assumptions. This is not surprising: Symbols that stand for book characteristics are the assumption contents of this type. A symbol for a book characteristic A can be interpreted as proposition "a book shows the characteristic A". Such a proposition can be negated; hence negative graduations are possible for the SBUPref* assumptions of Grundy.

3.1.3 Propositional Calculus for User Modeling: Complex Propositions

Huang's Student Model Maintenance System (SMMS) [Huang *et al.*, 1991] is the only system that we found to use complex formulas of propositional calculus. The main focus of the SMMS is the application of belief revision techniques in student modeling. Most work on belief revision has considered propositional calculus only; SMMS is no exception. Like most of the systems mentioned above, SMMS models user (i.e. student) knowledge of the teaching domain. SMMS organizes the user model as a "default package network", corresponding to thematical subfields of the domain. The assumptions are represented by atomic propositions A (representing "the student knows the concept/fact/etc. A") and by negated atoms $\neg A$ (representing "the student does not know A"); they are contained in the default packages. Additionally, SMMS employs rules for inferring new assumptions from assumptions that were previously acquired. These rules are implications that follow the pattern

$$A_1 \land \ldots \land A_n \rightarrow [\neg]B$$

So, the rule antecedent is a conjunction of atoms, while the conclusion is an atom that may be negated or not.

Following our approach of separating assumption type from assumption content, SMMS assumptions belong to types SBUB and SB$\sim$UB. For the representation of assumption contents, non-negated atomic propositions are sufficient, since negation is expressed by the assumption type. Then, SMMS inference rules allow to infer SBUB or SB$\sim$UB assumptions from a set of SBUB assumptions (given in the antecedent conjunction). In the following, inference rules that relate different assumption types will be encountered quite frequently.

3.2 Predicate Calculus

3.2.1 Introduction

With propositional calculus, complex statements can be expressed by combining basic statements with logical connectives. However, in basic statements, objects of a given domain and relationships between such objects often play the central part. First-order predicate calculus (FOPC) is a formal means to express such relationships logically. Its basic building blocks are terms denoting objects of a domain, and predicates or relations ranging over these objects. Terms can be constructed from variables, constants, and functions mapping objects to other objects. The application of a predicate symbol on a number of terms is called an atom. As the name suggests, atoms are the basic formulas of predicate calculus. If the syntax of atoms is given, formulas of predicate calculus can be built using the construction rules of propositional calculus given in definition 3.1.

Definition 3.4 (Terms) *Terms are constructed from variable and function symbols. Arbitrary symbols are permitted as variables and functions. Then:*

1. *Variable symbols are terms.*
2. *If f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.*

Note that constants are 0-ary functions. An example for a term is

$$\text{converter}(\text{ms-word}(\mathit{version}), \text{html})$$

where '*version*' is a variable, and 'html', 'ms-word', and 'converter' are 0-ary, unary, and binary function symbols, respectively. Terms without variables are called *ground*.

Definition 3.5 (Formulas) *Any symbol is permitted as predicate symbol. Then:*

1. *If P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is a formula. Such a formula is called atom. Variables that occur in the argument terms are called free or unbound.*
2. *The formula construction rules of propositional calculus (Definition 3.1, 1–6) also apply to predicate calculus.*
3. *Let $A(x)$ denote a formula that contains the free variable x. Then, $\forall x A(x)$ (for all objects in the domain; universal quantification) and $\exists x A(x)$ (there is an object in the domain; existential quantification) are formulas. In such formulas, x is said to be bound by $\forall$ or $\exists$, respectively.*
4. *1–3 are the only ways to construct formulas.*

Formulas that contain ground terms only are themselves called ground. If no functions with arity greater than zero are used in the construction of formulas, i.e., if terms are restricted to variables and constants, then predicate calculus becomes equivalent to propositional calculus. The reason is that in this case the number of possible ground atoms is finite and therefore each atom could be replaced by a dedicated atomic proposition. However, also in this case predicate calculus offers notational advantages when relations between objects shall be expressed logically.

Also in predicate calculus, the truth value of formulas recurs to the truth value of atoms. The truth value of an atom $P(t_1, \ldots, t_n)$ depends on whether the n-tuple of domain objects given by its argument terms $t_1, \ldots, t_n$ is an element of the relation that is denoted by the predicate symbol P or not. The following definition of a predicate calculus interpretation is informal in the sense that is not explicitly based on algebraic structures that determine the meaning of predicates and functions in an interpretation. However, it is formal enough to make clear how formulas of predicate calculus are interpreted.

Definition 3.6 (Interpretation (Predicate Calculus)) *In predicate calculus, an interpretation $\mathcal{I}$ consists of an interpretation function I and a domain D (a set of objects).*

- *In an interpretation, every n-ary function symbol is mapped onto a concrete n-ary function ranging over D, and every variable symbol is mapped onto an object in D. Thus, every term t is interpreted as an object $I(t) \in D$.*
- *Every n-ary predicate symbol P is mapped onto an n-ary relation $I(P) \subseteq D^n$. Then, for an atom $P(t_1, \ldots, t_n)$:*
  $I(P(t_1, \ldots, t_n)) = true$ *iff* $\langle I(t_1), \ldots, I(t_n) \rangle \in I(P)$.
- *The interpretation of complex formulas (A), $\neg A$, $A \wedge B$, $A \vee B$, $A \rightarrow B$, and $A \leftrightarrow B$ is computed like in propositional calculus (see Definition 3.2).*
- *Let $I\langle x/o \rangle$ be the interpretation function that varies from I only in the domain object o being assigned to the variable x. Then*
  $I(\forall x A(x)) = true$ *iff* $I\langle x/o \rangle(A(x)) = true$ *for all* $o \in D$, *and*
  $I(\exists x A(x)) = true$ *iff* $I\langle x/o \rangle(A(x)) = true$ *for at least one* $o \in D$.

A is true in $\mathcal{I}$ iff $I(A) = true$.

3.2.2 Predicate Calculus for User Modeling: Ground Formulas

The most basic way of employing predicate logic is using ground formulas or even ground atoms only. Using such expressions, statements about concrete objects that are represented as logical constants or terms can be made. In a knowledge base, the set of employed constants and terms is always finite. As a consequence, each ground atom could be replaced by a unique proposition symbol, so that the expressive power of a knowledge base of ground formulas does not exceed that of a knowledge base which contains formulas of propositional calculus.

Plan, goal and domain representation in [Wu, 1991] One important and influential system that was developed in the area of natural language dialog systems was the "Unix Consultant" (UC) [Wilensky *et al.*, 1984; Wilensky *et al.*, 1986]. Within the UC project, a significant amount of work was spent on user (i.e., dialog partner) modeling. In work associated with the UC project, Wu presents a method for active acquisition of user model contents [Wu, 1991]. In his system, a plan recognizer attempts to identify the plan that the user pursues with the consultation dialog. If the results of this process are not sufficient for giving a helpful reply, the system will generate active acquisition goals that determine the system's further attempts to elicit information from the user. If such a goal is found to be useful for supporting the dialog, the dialog system will actively query the user for her plans and goals concerning the consultation dialog. The utility of active acquisition goals is evaluated along several dimensions with the help of decision-theoretic methods.

In an example, Wu shows how user plans and goals as well as domain information is represented. From the user question

> How do I get to the center of the bay?

the following plan information is extracted [Wu, 1991, p.167]:

```
user-has-plan(plan1)
plan-has-goal(plan1,goal1)
user-knows-route-goal(goal1,location1,location2)
here(location1)
center-of(location2,bay1)
san-francisco-bay(bay1)
plan-composition(plan1,query-system1)
```

This is a set of ground atoms. Constants denote objects of the domain (like `bay1` and `location2`) but also plan representation elements (like `goal1` and `plan1`). Correspondingly, predicates are used to make statements about domain objects (e.g., `center-of` and `san-francisco-bay`) or about plan elements (`plan-has-goal`). The atoms that are built with these predicates express domain knowledge of the system. Therefore, they are SB-type assumptions. Other expressions represent assumptions about the user. The predicate `user-has-plan` concerns user plans (and, indirectly, goals), while `user-knows-route-goal` concerns user knowledge. These predicates are used to express SBUW and SBUB assumptions, respectively. In contrast to domain predicates, these "user model" predicates convey assumption type information. In section 3.2.4, several examples of user modeling work will be presented, where first-order predicates are more explicitly used to express assumption type information.

REPLAI-II REPLAI-II [Retz-Schmidt, 1991] is another plan recognition system. It operates within the system SOCCER, the aim of which is to generate descriptions of animated scenes in soccer games. REPLAI-II is to detect plans and intentions of agents that participate in a scene, e.g. players of both teams or the referee. Furthermore, interactions between agents, plan failures, and reasons for these failures shall be determined. In the observed scenes, the system tries to focus on specific events and agents to avoid information overload. However, the set of plan hypotheses may still be insufficiently constrained, even after several successive scenes have been observed. In order to select a best hypothesis, agent-specific information is considered. Like in the predecessor system REPLAI, plan preferences and interaction preferences are modeled using simple expressions with plausibility values[2]. A plan preference denotes the plausibility of an agent carrying out a certain plan. An example is

```
(plan_pref player7 individual_attack 0.9)
```

This is a Lisp-like notation of a ground atom, extended with a plausibility value. `player7` and `individual_attack` are constants denoting an agent and a plan, respectively. Like some of the predicates used by Wu (see above), the predicate `plan_pref` carries information about what is modeled, namely a plan preference. Interaction preferences are represented similarly. An example is

```
(interact_pref player7 player18 0.8)
```

which rates the plausibility tat `player7` prefers to interact with `player18` as 0.8.

The above expressions are not contents of a user model, but can be regarded as elements of an agent model, namely a model of `player7`. So, they express assumptions about an agent, but it is not obvious how they can be characterized in terms of assumption types. The first possibility is to distinguish two types for each agent, namely assumptions about plan preferences and assumptions about interaction preferences. Then, assumption contents would be the symbols that denote either plans or other agents, with attached plausibility values. The second possibility is to have only one type, namely assumptions about the agent's preferences, and take a predicate calculus notation to distinguish between plan preferences and interaction preferences. Let SB*player7Pref denote the suggested single assumption type for graduated assumptions about the preferences of agent `player7`. Then, the above examples could be denoted as assumption type expressions

[2]REPLAI-II does not represent relative preferences in the sense that an agent prefers A over B, but absolute preferences in the sense that A is among the preferred things of an agent.

SB*player7Pref : pursue-plan(individual-attack) [0.9]
SB*player7Pref : interact-with(player18) [0.8]

Goal representation in TRACK Another author who deals with the plan recognition problem of identifying user plans and goals in the context of a natural-language dialog system, namely the system TRACK, is Carberry. In [Carberry, 1989], she focuses on inferring assumptions about the user's focus of attention from one individual utterance, which are represented as a set of *candidate focused goals*. Each candidate goal is evaluated considering the dialog context. Heuristics are used to select the goal that is most appropriate to the existing dialog context. The selected goal is then incorporated into the model of the user's current plan.

The application domain of TRACK is consultation dialogs concerning college courses. For the representation of a plan in this domain, a STRIPS-like plan structure is used, which contains applicability conditions, preconditions, a plan body, and effects of the plan. A plan is identified with a goal, and the plan body is a set of subgoals that are to be achieved in order to accomplish the plan. The subgoals are constrained by further conditions. In STRIPS [Fikes and Nilsson, 1971], logic-based reasoning was used for planning. Therefore, it is not surprising that goals, subgoals, and subgoal constraints are represented in TRACK as atoms of predicate calculus. Consider the following example:

Learn-From-Person(_agent:&Person, _sect:&Sections, _fac:&Faculty)

is a subgoal in a domain plan concerning the goal of an agent to learn a course section that is taught by some member of the faculty. The symbols '_agent', '_sect' and '_fac' are logical variables. Their possible values are constrained by the sort specifications '&Person', '&Sections' and '&Faculty'. During the plan recognition process, variables may become substituted by constants. This is a standard step of a logical reasoning procedure. However, plan recognition in TRACK is realized by a specialized algorithm and not by a general logical inference engine.

Despite of the use of variables, representation and reasoning in TRACK semantically does not exceed the limits of propositional calculus. Similar to STRIPS, variables can only be substituted by a finite number of values. The positive effect of this is that reasoning remains decidable. Nevertheless, TRACK can be considered an example of the sensible use of predicate logic syntax for structuring propositions into relations and objects. However, the use of logical terms as predicate arguments is restricted. E.g., the constant 'FRENCH112-10-FALL' denotes section 10 of course French 112, taught in the fall. This could alternatively be expressed with a complex term like 'section(FRENCH112,10,FALL)'. Thus, information about the section object would be more structured. This could be useful if the three main elements of a section object (course identifier, section number, and teaching term) were to be handled separately in reasoning processes.

3.2.3 Predicate Calculus for User Modeling: Rules and Complex Formulas

ANATOM-TUTOR ANATOM-TUTOR [Beaumont, 1994] is a tutoring system in the domain of anatomy. It comprises an adaptive hypertext component and an adaptive component for simulating examination situations. Both components take the assumed knowledge of the user about the application domain into account. An ANATOM-TUTOR user model contains the domain concepts that the user is assumed to know, which may be objects and functions as well as object and function classes of human anatomy. These concepts are represented by their names, i.e., as propositional atoms. Moreover, relations between concepts may also be assumed to be known to the user, like 'is-part-of' relationships between anatomy objects. Such relationships correspond to ground atoms of predicate calculus. Finally, the user may be familiar with laws of anatomy. In [Beaumont, 1994], only an informal example for such a law is presented, namely

> "any ganglion which receives nerve-impulses from a parasympathic object is itself parasympathic" (cf. p.33)

This is rule-like information, which is represented as a production rule in the formalism of the rule-based programming language Ops5. A concrete representation of such a domain law is not presented in [Beaumont, 1994], but Ops5 production rules are closely related to implicational expressions of FOPC. In the paper it is stated that the relations 'belongs-to-class' and 'sends-nerve-impulses-to' are used. Using corresponding predicates, the above rule could be represented in predicate calculus as follows:

$$\begin{array}{rl} \forall x, y & \text{belongs-to-class}(x, \text{ganglion}) \\ & \wedge\ \text{belongs-to-class}(y, \text{parasympathic-object}) \\ & \wedge\ \text{sends-nerve-impulses-to}(y, x) \\ \rightarrow & \text{belongs-to-class}(x, \text{parasympathic-object}) \end{array}$$

This FOPC formula is an implication with a conjunction of atoms as antecedent and a single atom as conclusion. Such implications are Horn rules, equivalent to Horn clauses that are the basis of Prolog. Horn rules are sufficient for many representational purposes.

An interesting aspect of ANATOM-TUTOR is that the domain rules are part of the user model (assumption type: SBUB) and not part of the domain model of the system. They can be used to infer additional assumptions about the user by simulating the derivations she can perform according to the rules that she is assumed to know.

Quilici's explanation rules In [Quilici, 1994], methods are presented for inferring assumptions about the user from her feedback to statements or explanations of an advisory system. The modeled assumptions concern the plan-related knowledge of the user. Like in earlier work [Quilici, 1989], Quilici introduces a set of basic relations for representing planning knowledge. He distinguishes between factual relations for expressing standard planning relationships and evaluative relations for expressing subjective opinions about the relative or absolute desirability of a plan or action.

Every factual relation has a corresponding negative relation. An example is [Quilici, 1994, p.330]

results(E, E')	E' is a consequence of executing E
not-results(E, E')	E' is not a consequence of executing E

These relations are first-order predicates, with events or effects E as arguments. Since Quilici explicitly states that an instantiated relation is contradictory with its negation, 'not-results(E, E')' could be represented as '$\neg$results(E, E')' in FOPC.

The basic evaluative relation 'desirable' is a one-place predicate applied to single events; also for 'desirable' there is a negative relation 'not-desirable'. Two other evaluative relations relate events E and goals G:

most-desirable(E, G)	E is the best way to achieve G
more-desirable(E, E', G)	E is better than E' for achieving G

Furthermore, Quilici introduces *explanation rules* for deducing evaluative and factual beliefs. He describes them as 'IF...THEN...' rules, for instance:

IF results(E, E') AND desirable(E') THEN desirable(E)

(An event is desirable if it has a desirable effect.)

Logically, explanation rules correspond to first-order implications. The rule above corresponds to

$$\forall E, E' \, \text{results}(E, E') \land \text{desirable}(E') \rightarrow \text{desirable}(E)$$

This example is a Horn rule, like all other rules for inferring 'desirable' and 'not-desirable' statements. However, for relative evaluations like 'more-desirable' and 'most-desirable', more complex formulas are needed. Quilici introduces an additional 'not-exists' relation for these purposes:

IF	results(P, G) AND results(P, E) AND desirable(E) AND not-exists$(P'$, results(P', G) AND results$(P', E))$
THEN	most-desirable(P, G)

(A plan is the most desirable plan for a goal if it is the only plan that achieves the goal.)

Logically, 'not-exists' is no predicate, but naturally corresponds to a negated existential quantifier:

$$\begin{array}{lrl} \forall P, G, E & & \text{results}(P, G) \wedge \text{results}(P, E) \wedge \text{desirable}(E) \\ & & \wedge \neg\exists P' \ (\text{results}(P', G) \wedge \text{results}(P', E)) \\ & \rightarrow & \text{most-desirable}(P, G) \end{array}$$

Planning knowledge can be attributed to both the user and the system. Also, rules can be applied for inferences concerning both agents. See Section 3.2.4 for a discussion of these aspects of Quilici's work.

Acquisition rules in GUMAC For his system GUMAC, Kass developed a set of acquisition rules for making default inferences about user's beliefs from their interaction with an advisory expert system [Kass, 1991]. These rules are of a heuristic nature, and they are implemented procedurally and not with a general logical mechanism. However, in [Kass, 1991] they are described in a complex logical formalism and give an example how complex logical formulas could be employed. The basic knowledge representation of GUMAC deals with concepts of a domain terminology and with goals of system and user, which can be satisfied if a corresponding action is performed. GUMAC can represent assumptions about the user's beliefs concerning terminology, goals or actions, but also assumptions about what the user believes the system to believe. In general, GUMAC does not only deal with basic user modeling, but with an agent modeling scenario with user and system being interchangeable agents.

There are two types of acquisition rules in GUMAC: *Local rules* are considered to be inference rules held by the agent being modeled; they can be used to infer further beliefs of this agent by simulating its reasoning. *Applied rules* are also part of the model of one agent (called "primary agent" (PA)), but are used to draw conclusions about what beliefs other agents ("secondary agents" (SA)) are assumed to hold. All rules are described as default rules, using the symbol $\leadsto$ to denote a default implication. GUMAC employs a truth maintenance system to implement retractability of default assumptions.

Three local rules are introduced, which mainly describe standard inferences of concept-based knowledge representation. An example is the so-called "concept transitivity rule"

$$\text{isa}(A, B) \wedge \text{isa}(B, C) \leadsto \text{isa}(A, C)$$

where 'isa' represents a subclass relation between concepts. If it is present in a user model, the user is assumed to know that the subclass relation is transitive. The rule holds for all concepts A, B and C, so that in a classical FOPC notation, quantifiers would have to be added:

$$\forall A, B, C \ \text{isa}(A, B) \wedge \text{isa}(B, C) \rightarrow \text{isa}(A, C)$$

In order to make statements about a secondary agent's beliefs in applied rules, Kass employs the notation $\mathrm{Bel}(SA, p)$, where p may be any proposition. Applied rules can be very complex: on both sides of the implication $\rightsquigarrow$, conjunctions and disjunctions appear, and simple propositions as well as $\mathrm{Bel}(SA, p)$ expressions may be negated. One of the less complex examples is the "expert rule"

$$\mathrm{Bel}(SA, \mathrm{expert}(PA)) \wedge \mathrm{do}(PA, \mathrm{tell}(PA, SA, P)) \rightsquigarrow \mathrm{Bel}(SA, P)$$

According to Kass, this rule is reasonable when SA is the (novice) user, and PA is the system. Then the rule states that the user believes what the system tells her, if she believes the system to be an expert. Since the rule holds for all propositions p, and even for arbitrary SA and PA combinations, in a classical notation universal quantifiers would be added. Unlike the local rule above, this rule could be regarded as a second-order rule scheme, since it contains quantified variables for propositions. However, within the scope of the 'Bel' and 'do' predicates, propositions are terms that can be replaced by normal quantified variables. Predicates like 'Bel' are employed quite often in user modeling systems, so that the representation of assumption contents as terms is not unusual. Several more examples of this kind of representation will be discussed in the following section.

3.2.4 Using Predicates for Assumption Type Information

In the previous section, several user modeling systems and approaches were presented, the representation facilities of which explicitly use or can be characterized with predicate calculus expressions. In some of these systems, predicates were used that carry information on the kind of knowledge, i.e. on the type of assumption the expression makes a statement about. Such "assumption type predicates" have been used in a great number of user modeling work. This section presents several more examples, which seem to integrate assumption type and content information in one formalism. However, in most of these systems, type and content information can be easily separated, as was already demonstrated for REPLAI-II.

Goal, belief, and competence modeling In their approach to natural language text generation, [Moore and Paris, 1992] employ a user model. It contains assumptions about user knowledge, goals and competence. The user may be assumed to know about domain concepts and their relationships, but also to know about problem-solving actions. Possible competences of the user are to perform an action or to achieve a goal. The expressions that are contained in a user model are instances of one of the following patterns:

- `(GOAL USER` *Goal*`)`
  The user is assumed to have a certain *Goal*.
- `(BEL USER` *P*`)`
  The user is assumed to believe in a proposition *P*. Propositions may refer

to domain concepts, like `(CONCEPT` *C*`)` or `(ISA` *C1 C2*`)`, or to possible actions and goals of the user, like `(STEP` *Action Goal*`)`.

- `(COMPETENT USER` *Comp*`)`
  The user is assumed to have the competence *Comp*. Two kinds of competence expressions are possible, namely `(DO USER` *Action*`)` and `(ACHIEVE USER` *Goal*`)`.

All these expressions can be regarded as FOPC formulas. They correspond to ground atoms, with predicates 'goal', 'bel', and 'competent' for assumption type information, and with functions like 'concept', 'isa', 'step', 'do', and 'achieve' for the construction of terms that denote assumption contents. Assumption type and content information is clearly separated, so that user model contents can easily be rewritten using our notation for assumption type expressions. Let the types of the above assumptions be denoted with SBUW for goals, SBUB for believes, and SBUC for competences. Then, the assumptions can be expressed as

SBUW:$Goal$
SBUB:concept(C), SBUB:isa($C1$,$C2$), SBUB:step($Action$,$Goal$)
SBUC:do($Action$), SBUC:achieve($Goal$)

The content terms from above have become content formulas, the functions have become predicates. Note that the 'user' parameter of 'do' and 'achieve' could be omitted because of its redundancy.

In the above example, assumption contents are represented as simple FOPC formulas. However, a closer look at the content expressions shows that different, specialized formalisms might be used to represent assumption contents more appropriately. 'concept' and 'isa' expressions could be handled by a semantic network formalism, a KL-ONE like system or a more modern description logic. Actions and goals could be represented in a graph-based formalism for actions and plans like those of [Cohen *et al.*, 1991; Goodman and Litman, 1992]. See Section 3.3 for a discussion of specialized representation formalisms. However, this example indicates that such specialized formalisms will often be *logic-based* in the sense that their structures correspond to expressions of classical logic.

Quilici's representation of planning knowledge In two papers, Quilici deals with the generation of cooperative explanations in advisory systems [Quilici, 1989; Quilici, 1994]. The cooperativity is mainly based on a model of both the user's and the system's (the "advisor" in Quilici's words) plan-related beliefs. In Section 3.2.3, we discussed the representation of planning knowledge and of "explanation rules" for inferring planning knowledge, as presented in [Quilici, 1994]. In [Quilici, 1989], similar representation means are used. In both papers, the central representation facility is a predicate

$$\text{belief}(B, R)$$

where B is either the 'user' or the 'advisor', and R may be a piece of planning knowledge. In [Quilici, 1989], it is stated that a 'belief(B, R)' expression "represents an advisor's belief that an agent maintains that a particular plan applicability condition, precondition, or effect holds. The agent is either the user or the advisor" [p.112]. Hence, 'belief(B, R)' expressions correspond to SBUB and SBSB assumptions. However, the reflexivity in SBSB seems redundant; expressions of the form 'belief(advisor, R)' can also be regarded as SB assumptions.

Explanation rules (see Section 3.2.3 for examples) can be instantiated for backward deductions concerning both user beliefs and advisor beliefs. In order to express this formally, all atoms A in explanation rules would have to be replaced by expressions 'belief(B, A)'. Since the belief predicate only distinguishes two agents, and since it must not be nested, Quilici's representation corresponds to two separate knowledge bases for SB and SBUB assumptions. However, explanation rules would have to be present in both knowledge bases in order to be available for both user-related and advisor-related inferences. In a partition approach like that of [Kobsa, 1990] with inheritance relations between partial knowledge bases, the rules could be placed into a partition for "shared beliefs" that would inherit to two other partitions, one for SB and the other for SBUB assumptions (cf. Section 2.3.3). The corresponding "shared beliefs" assumption type would be useful anyway, since it is important for the utterance generation in Quilici's system to know whether a belief is shared between user and advisor.

First-order rules for weighted abduction [Appelt and Pollack, 1992] present weighted abduction as a method for ascribing a plan to the user based on current observations. In the plan recognition context, abduction means ascribing to an agent a set of beliefs and intentions that explains or entails a given observation. On principle, weighted abduction is an attempt to prove the observation with a set of rules. Costs are associated with rule conclusions, and (cost) weights are associated with antecedents. Subgoals of the backward reasoning process may be assumed without further proof. Such assumptions cause weighted costs, while a real proof does not cost anything. The "cheapest" set of assumptions that need to be made to prove the observation is the solution of the abduction problem.

[Appelt and Pollack, 1992] use a representation formalism developed in [Konolige and Pollack, 1989]. The basic elements of this formalism include predicates ("epistemic operators" in the words of Appelt and Pollack) to describe an agent's mental state, which are defined as follows [Appelt and Pollack, 1992, p.12]:

$\mathrm{INT}(a, \alpha)$	agent a intends α
$\mathrm{BEL}(a, p)$	agent a believes p
$\mathrm{ACH}(a, p)$	agent a believes p will become true as a consequence of the action he performs ("achieves")
$\mathrm{EXP}(a, p)$	$\mathrm{BEL}(a, p) \lor \mathrm{ACH}(a, p)$ ("expects")

Latin letters like p denote factual propositions, while Greek letters like α denote actions. Complex actions (i.e., plans) are constructed with the operators TO and BY:

$\mathrm{TO}(\alpha, p)$	the plan consisting of doing α to make p true
$\mathrm{BY}(\alpha, \beta, p)$	the plan consisting of doing β by doing α, while p is true (p "enables" the relation)

We will see that the epistemic operators are first-order predicates and both propositions and actions are represented as first-order terms.

Using the above facilities, plan ascription rules (or better, rule schemes) can be formulated. An example is

$$\mathrm{BEL}(a, \mathrm{TO}(\alpha, p)),\ \mathrm{INT}(a, \alpha),\ \mathrm{ACH}(a, p)\ \xrightarrow{to}\ \mathrm{INT}(a, \mathrm{TO}(\alpha, p))$$

The commas on the left hand side of the rule arrow can be read as conjunction symbols. The rule scheme says that if an agent believes an action to have an effect, intends to do the action and to achieve the effect, he intends to do the action in order to achieve the effect. For integrating ascription rules into the weighted abduction mechanism, they must be converted into a logical format. Appelt and Pollack present the conversion of the above ascription rule as example (a slightly different notation is used here):

$$\forall a, \alpha, p\, \mathrm{BEL}(a, \mathrm{TO}(\alpha, p)) \wedge \mathrm{INT}(a, \alpha) \wedge \mathrm{ACH}(a, p)\ \rightarrow\ \mathrm{INT}(a, \mathrm{TO}(\alpha, p)) \quad (3.1)$$

Appelt and Pollack themselves call this a rule (not a scheme). Since this rule quantifies over actions and propositions, action and proposition expressions must be terms and not formulas. However, action and proposition terms can be regarded as proposition-valued. This is partly in line with the syntactic approach to reasoning about beliefs, which was mainly pursued by Konolige [Konolige, 1982] (but see also [Haas, 1986], or see [McArthur, 1988] for an overview). In this approach, first-order predicates like 'fact' (for objective facts) and 'bel' (for beliefs) are elements of a meta-level language. The terms of such a meta-language are names of formulas of an object-level language, which is used to express the belief contents. So, the values of meta-language terms are object language formulas. Reasoning in the syntactic approach is very expensive, if the object language allows full expressiveness of, say, propositional calculus with logical connectives. The reason is that object language reasoning must then be simulated in the meta language, which involves much more steps then object reasoning itself. In the weighted abduction rules of Appelt and Pollack, an object language for both actions and factual propositions would not exceed FOPC atoms, so that object language reasoning is obsolete. Hence, action and proposition expressions can be reduced to terms of a language with assumption type predicates without loss of reasoning possibilities. So, a distinction between meta-language and object language is not necessary.

An important property of formulas like the weighted abduction rule (3.1) is that they cannot be split into assumption type and content. Different assumption type predicates are used within one formula, so that a relationship between different assumption types, namely beliefs, intentions and achievements, is expressed. This already indicates that a pure assumption type approach is of limited expressiveness and that an extension is needed. However, a distinction between assumption types and contents has its advantages, which may become more clear in the next section.

Summary and Outlook The use of predicates for assumption type information was led by the desire to use distinct means of a formal language to state that something is believed, intended, wanted, etc. In section 3.4, *modal logic* will be presented as formalism that introduces separate operators for characterizing logical formulas. These modal operators are generic; they can be parameterized if different kinds of characterizations shall be represented and if formulas are to be associated with different agents.

3.3 Specialized Logic-Based Formalisms

3.3.1 Introduction

In the previous sections, propositional and predicate calculus were introduced. Because of their generality, such logical formalisms can be employed in many different application areas. However, their generality is also responsible for their inherent inefficiency. For specific representation and reasoning tasks, specialized formalisms will probably be more appropriate than the general logical languages.

This section will describe specialized knowledge representation techniques that can be and have been employed in the area of user modeling. We will review formalisms that were mainly developed to achieve a better structuring of knowledge, and that turned out to be especially suitable to representing knowledge about the concepts of a domain. Most of these mechanisms are logic-based; their language constructs, i.e. the meaning of these constructs, can be described in terms of predicate calculus. Unlike FOPC, however, concept-based formalisms have been employed for representing assumption contents only. In most cases, an assumption type was assigned to concept-based representations by storing them in a dedicated knowledge base.

Early knowledge representation formalisms like frames [Minsky, 1975] and semantic networks [Quillian, 1968] were introduced mainly to allow better structuring of knowledge. Frames were developed by Minsky to represent prototypical situations and objects. Their basic structure corresponds to records of standard programming languages, with attribute-value pairs as building blocks. Minsky criticized the use of logic for knowledge representation, but it could be shown that the main aspects of frames only syntactically differ from what can be expressed in predicate calculus.

Semantic networks were supposed to represent the meaning of natural language. Essentially, a semantic network is a graph. Its nodes may denote concepts (i.e., classes of objects) or objects of the world. Relationships between nodes can be represented with property links and IS-A links; the former represent properties of concepts and objects, while the latter represent hierarchical relationships between concepts or instantiation relations between objects and concepts.

Semantic networks lack a formal semantics. The meaning of a specific network depends dramatically on the underlying implementation. An example for a confusing element is the IS-A link. An IS-A link between two concept nodes C_1 and C_2 states that all objects of class C_1 are also objects of class C_2. In predicate calculus, this could be expressed with the formula

$$\forall x \, C_1(x) \rightarrow C_2(x)$$

However, if it is drawn between an object node o and a concept node C, an IS-A link means that the object o is an instance of class C, which simply corresponds to

$$C(o)$$

in FOPC.

An important move towards formalisms with a well-defined formal semantics for representing concepts, objects and their relationships was made with the introduction of the structured inheritance networks of the KL-ONE system [Brachman and Schmolze, 1985]. In KL-ONE and its many successors, concepts and roles are the basic elements, which correspond to one-place and two-place predicates, respectively. Furthermore, a set of well-defined constructors allow the definition of complex concepts and roles. The semantics of these representation elements can be described in a set-theoretical way or by translating them into predicate calculus. Since this is possible for all presently known constructors, KL-ONE-like terminological representation formalisms can be regarded as specialized variants of FOPC. In early KL-ONE-like systems, concept hierarchies or terminological knowledge bases were syntactically represented as graph structures. In the last years, the formal, logical character of terminology representation has been more important, so that the formal languages of *description logics* have become more prominent.

Table 3.1 presents some examples for concept definition terms that are common in description logics [Baader, 1996]. In the left column, an interface syntax for concept terms is presented that could be used in an implementation. The middle column uses the standard abstract syntax for description logics, as it is used in theoretical work. The right column then contains a translation of each concept term into predicate logic. In the translations, there is a free variable x. This variable gets bound, when a concept term CT is used in the definition of a concept C. The standard abstract syntax for concept definitions is $C \sqsubseteq CT$, and a possible interface syntax is $(\mathsf{isa}\ C\ CT)$. The FOPC translation of a concept definition is $\forall x\,(C(x) \rightarrow CT)$; in this formula, the free variable x of CT gets bound. $C \sqsubseteq CT$ is called a partial definition [Owsnicki-Klewe *et al.*, 1993]; a full definition is $C = CT$, meaning $\forall x\,(C(x) \leftrightarrow CT)$. A possible interface syntax for a full definition is $(\mathsf{equal}\ C\ CT)$. Similar to concepts, also roles can be defined. A description logic knowledge base (strictly speaking: the terminological part of a description logic knowledge base, also called *TBox*) is a set of concept and role definitions.

interface syntax	*descr. logic*	*FOPC translation*
C	C	$C(x)$
$(\mathsf{and}\ C_1 \ldots C_n)$	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$
$(\mathsf{or}\ C_1 \ldots C_n)$	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$
$(\mathsf{not}\ C)$	$\neg C$	$\neg C(x)$
$(\mathsf{some}\ R\ C)$	$\exists R.C$	$\exists y\ R(x,y) \wedge C(y)$
$(\mathsf{all}\ R\ C)$	$\forall R.C$	$\forall y\ R(x,y) \rightarrow C(y)$

Table 3.1: Syntax and semantics of concept terms in description logics

The concept terms of Table 3.1 can be read as follows:

- C is an atomic term, which denotes all objects of class C;
- $(\mathsf{and}\ C_1 \ldots C_n)$ is the intersection of classes $C_1, \ldots, C_n$;
- $(\mathsf{or}\ C_1 \ldots C_n)$ is the union of classes $C_1, \ldots, C_n$;
- $(\mathsf{not}\ C)$ is the class of objects that do not belong to class C;
- objects of class $(\mathsf{some}\ R\ C)$ are required to be related to one object of class C via the relation (the attribute or role) R;
- and for objects of class $(\mathsf{all}\ R\ C)$, all values of attribute R must be objects of class C.

3.3.2 Concept-Based Representation

Many user-adaptive systems individualize their behavior according to the user's knowledge about the concepts of a domain. For instance, explanation generating systems will care for the user knowing the concepts that are used for an explanation. Information systems like adaptive hypertexts may tailor the amount and detail of provided information to the conceptual knowledge of the current user. So, formalisms that are specialized on the representation of concepts, concept hierarchies, or more complex terminologies are employed quite frequently in user modeling systems. Often, concept hierarchies are predefined as system domain knowledge, and user model contents are relative to this knowledge. The use of concept-based formalisms in user modeling knowledge bases is the focus of this section.

However, there are also some examples for the utilization of frame-like representations. [Lehman and Carbonell, 1989] present the adaptive parser CHAMP, which acquires an individual grammar via the interaction with a user, in order to adapt its own linguistic behavior to the user's idiosyncrasies. So, the user model contains syntactic forms that describe the individual grammar of the user. The example domain of CHAMP are the activities of an executive assistant. For representing objects and actions of this domain (i.e., for SB assumption contents) a frame formalism is used. A frame-like, object-oriented representation with attribute-value pairs is also used in the plan-based help-system TECHDOC-I [Peter and Rösner, 1994]. In this system, both a plan hierarchy and an object hierarchy, which can be regarded as SB-type knowledge, are represented in this formalism. Dialog history information is integrated by having an attribute in both plan and plan step frames, the value of which indicates how often a user has already executed a plan or a plan step, respectively. A model of the user's knowledge is represented with "degree of familiarity" (DOF) attributes in object frames. This representation corresponds to SBUB* assumptions, the contents of which can be represented as propositional atoms (the object names), while the DOF values are graduation values (cf. Section 3.1).

A first example for the use of conceptual hierarchies is the early user modeling system UMFE [Sleeman, 1985]. UMFE employs the concept database of its host system NEOMYCIN, an expert system for bacterial infections, which can be regarded as a knowledge base of SB assumptions. NEOMYCIN concepts are infectious processes, which are structured in a tree-like classification hierarchy. With each concept, two numerical values are associated that represent the importance and the difficulty of the concept, relative to the knowledge of a hypothetical second-year medical school student population. A small excerpt of the NEOMYCIN database, as it is presented by Sleeman, is

```
INFECTIOUS-PROCESS 10 2
   HEAD-INFECTION 7 2
      OTITIS-MEDIA 8 5
      SINUSITIS 6 5
```

where in each line the first number is the importance value, and the second number is the difficulty value. In terms of semantic networks, a hierarchical relation in NEOMYCIN corresponds to an IS-A link, e.g. 'OTITIS-MEDIA IS-A HEAD-INFECTION', while the two numbers can be seen as end nodes of property links named 'importance' and 'difficulty'. Hence, also an FOPC formulation is possible:

$$\forall x \text{ OTITIS-MEDIA}(x) \rightarrow \text{HEAD-INFECTION}(x)$$

would represent hierarchical subordination, and formulas like

$$\forall x \text{ OTITIS-MEDIA}(x) \rightarrow \text{importance}(x, 8)$$

would link concepts to importance and difficulty values. However, the tree-like representation of NEOMYCIN is absolutely sufficient and perfectly suitable for the purposes of UMFE. UMFE models the user's knowledge of the concepts in the database. With each concept, it associates a value that expresses the strength of the assumption that the user knows the concept, thus representing SB*UB assumptions (cf. Section 3.1.2). Furthermore, both hierarchical relationships and numeric values are used for inferences that UMFE makes concerning user knowledge (cf. Section 4.1.3).

[McCoy, 1989] deals with generating responses to misconceptions which can be detected in a user's behavior. In particular, her system treats what she calls object-related misconceptions. So, both the system's model of the world (SB assumptions) and the user model (i.e., the assumed user's model of the world, SBUB assumptions) contain "an object taxonomy with attribute-value pairs attached to the objects". McCoy does not describe this representation in detail, but she uses the following semi-formal notations to talk about system and user knowledge that make the close relationship to terminological representations clear:

X is-a Y
X is-not-a Y
X has attribute A with value V

where X and Y stand for concepts of the taxonomy. In addition, a number of service functions of the representation formalism are exploited, like 'Type-of(X)' and 'attributes-of(X)'. Note that McCoy stores SB and SBUB knowledge in separate knowledge bases, in accordance with the assumption type representation approach.

In section 3.2.4, the representation formalism of [Moore and Paris, 1992] was presented as an example for the use of first-order predicates to express assumption type information. Moore and Paris use expressions like

`(CONCEPT` *C*`)`
`(ISA` *C1 C2*`)`

as terms in their formalism to represent the contents of SBUB assumptions. With these expressions, elements of concept hierarchies or of semantic networks can be described. The meaning of the `ISA` expression is immediately clear; an FOPC correspondence for IS-A links has been presented above. As SBUB assumption content, a `CONCEPT` expression is to represent that the user knows *about* a concept, not that she knows the concept in the sense of knowing all properties of the concept and other related items. An FOPC correspondence of such expressions is not immediately clear. At first sight, `(CONCEPT` C`)` could be translated straightforward to 'concept(C)' (cf. Section 3.2.4). But usually, concept names correspond to one-place predicates in FOPC transformations of semantic networks, so that they cannot become arguments of a 'concept' predicate. This problem can be solved, if a universal "top" or "thing" concept is available in the hierarchy, which all other concepts are subconcepts of. Then, `(CONCEPT` C`)` can be regarded as abbreviation of `(ISA` C `thing)`, which has the usual FOPC correspondence. Thus, a `CONCEPT` expression simply states that a specific concept is available or present in a knowledge base. Then, a complete user model expression

`(BEL USER (CONCEPT` C`))`

can be interpreted as the user knowing that there is a concept C.

Also work of Kass was presented in Section 3.2, who employs a complex logical language to formulate the user model acquisition rules of his GUMAC system [Kass, 1991]. In this language, he uses a set of representational primitives, which divide into definitional knowledge primitives and strategic knowledge primitives. There are three definitional primitives, which, according to Kass, describe concepts, their properties, and their relations.

concept(C)	C is a concept
property(C, P)	concept C has a property P
isa($C1, C2$)	concept $C1$ is a subclass of $C2$

The structures that can be described with these primitives clearly correspond to those that are possible in semantic networks. However, like in [Moore and Paris, 1992], the 'isa' primitive is only applicable to concepts, since individual objects are not represented. So, the typical "IS-A confusion" of semantic networks is avoided. Actually, the definitional primitives only describe the domain representation in the formal notation of acquisition rules. In the system GUMAC, the terminological representation system NIKL, which is a KL-ONE implementation, is used to represent both the user's beliefs about the domain and the domain knowledge base of the system. However, as it is indicated by the definitional primitives, only a subset of the facilities of NIKL is employed.

Kass makes a statement about the intended meaning of the 'concept' primitive. In accordance with our interpretation of the `(CONCEPT` C`)` expressions of [Moore and Paris, 1992], a complete belief assertion

$$\mathrm{Bel}(SA, \mathrm{concept}(C))$$

where the "secondary agent" SA normally is identified with the user, "does not imply that SA knows all about C. Rather, it implies only that SA knows something about C (has a concept for it)" [Kass, 1991, p.233].

There are still more systems that utilize concept-based representation formalisms, which cannot be described in detail here. An early example is the system VIE-DPM [Kobsa, 1985], which was described in Section 2.3.2 as representative of the partition approach. Two further examples shall be mentioned briefly. [Sarner and Carberry, 1992] employ a KL-ONE-like structured inheritance network to represent domain knowledge (i.e., SB assumptions) in a system for generating user-tailored definitions of domain concepts. For definition generation, assumptions about the user's familiarity with domain concepts are crucial. They are represented by storing a copy of the domain knowledge base in the user model and assigning numerical belief factors to each representation entity. Thus, SB*UB assumptions are realized. The second example is the system KN-AHS [Kobsa *et al.*, 1994], which shall be mentioned as a representative of the class of adaptive hypertext systems. KN-AHS adapts the amount and the detail of information that is presented on hypertext pages to the user's assumed knowledge about the concepts that occur in the text. It uses a KL-ONE formalism for representing the domain terminology with concepts and roles. The adaptation algorithms of KN-AHS require explicit representation of the concepts the user is assumed to know as well as the concepts the user is assumed to not know. Hence, separate SBUB and SB$\sim$UB collections of concepts are maintained.

3.3.3 Other Specialized Formalisms

There is a wide range of methods that have been employed for user model representation and reasoning. In particular, numeric (e.g., probabilistic) methods have become widely used in the last years. However, there are also further specialized representation mechanisms that may not be logic-based in a strict sense, but are related to logic-based representations due to their symbolic approaches.

Particularly in plan recognition systems, several formalisms have been employed for representing plan knowledge of the application domain, but also for representing the user's assumed knowledge about plans as well as her goals that indicate a current plan. In work of Carberry [Carberry, 1989; Sarner and Carberry, 1992], STRIPS-like definitions of planning operators with preconditions, applicability conditions, effects, and subplans or suboperators are employed in the domain knowledge base. Conditions and effects are represented in a FOPC-like formalism, as was demonstrated in Section 3.2. The plans and goals of the user are represented in a tree-like "context model", which is a structure of possible goal/subgoal sequences the user may pursue. One goal in the tree is marked as the current focus of attention.

In the GUMAC system, Kass employs a representation formalism for problem-solving knowledge that is utilized to model the current activities of the user and goal/subgoal relationships. Although in [Kass, 1991] a FOPC notation is employed to refer to actions and goals in the description of GUMAC's acquisition rules, Kass actually implements problem-solving knowledge using the specialized formalism EES (the "Explainable Expert System").

A different approach to plan representation is the use of graph-based formalisms to describe abstraction and decomposition hierarchies of plans and actions, as for example in [Goodman and Litman, 1992]. Abstraction and decomposition hierarchies can be logically formalized, as Kautz (see, e.g., [Kautz, 1991]) has shown. [Weida and Litman, 1992] attempt to implement such a plan representation with a KL-ONE-like formalism that is extended by means for representing temporal relationships.

In several intelligent tutoring systems, so-called "genetic graphs" are utilized for representing both static and dynamic aspects of the user's conceptual knowledge. [Niem *et al.*, 1993] present extended genetic graphs, with nodes representing learner knowledge about facts, concepts, rules and procedures of the tutoring domain, and links that implicitly represent the student's learning process. The semantics of the nodes and their possible meanings is characterized in a logic-based formalism, namely conceptual graphs, while the semantics of the links is described with predicate transition networks.

3.4 Modal and Epistemic Logic

3.4.1 Introduction

In a user model, assumptions or beliefs about the user are represented. These system beliefs will often be concerned about knowledge or beliefs of the user, but also about user goals, interests or preferences. In Section 1.3, assumption types have been introduced in order to allow a distinction of different kinds of assumptions. So, the assumption that the user wants or has the goal p was denoted by SBUW:p, meaning that the user model contains an assumption of type SBUW with assumption content p.

In the previous sections of this chapter, logic-based formalisms have been dealt with that were mainly used for expressing assumption contents p. In addition, we demonstrated how special predicates in FOPC formalizations were utilized for representing assumption type information. In the latter approaches, assumption type and assumption content information get mixed in one formalism. There are further logical means to handle assumption types and contents in one formalism, but with a higher degree of distinction between type and content information. In philosophy and artificial intelligence research, there has been a great amount of work on logics for reasoning about knowledge and belief, their origin being *epistemic logic*, which was developed by Hintikka [Hintikka, 1962; Hintikka, 1969].

In epistemic logic[3], special operators like B or K are introduced that characterize logical expressions as "believed" or "known", respectively.

Epistemic logic is a specific variant of modal logic. In modal logic, a necessity operator $\Box$ and a possibility operator $\Diamond$ are added to standard propositional or predicate calculus. As far as syntax is concerned, only one new construction rule for formulas needs to be added to the syntax definitions of either propositional (Definition 3.1) or predicate calculus (Definition 3.5):

Definition 3.7 (Modal logic syntax extension)

- *If A is a formula, then $\Box A$ and $\Diamond A$ are formulas.*

The standard reading for $\Box A$ is "it is necessary that A". The standard reading for $\Diamond A$ is "it is possible that A".

Modal logics usually have a *possible worlds semantics*, which was developed by Kripke [Kripke, 1963]. Intuitively, possible worlds are distinguishable states of affairs, which may be linked pairwise by an accessibility relation. If a possible world w' is accessible from a world w, this means that w' is an "imaginable" state of affairs if the actual state of affairs were w. The interpretation of non-modal expressions and, in modal predicate logic, of terms may differ between worlds. The truth of a modal expression $\Box A$ in a world w depends on the truth of A in the worlds that are accessible from w: It is necessarily true that A if and only if A is true in all imaginable (i.e., accessible) worlds. The actual state of affairs is represented by a designated world, and the overall truth of an expression in modal logic is defined by its truth in this actual world.

Definition 3.8 (Interpretation (Modal Logic)) *A modal logic interpretation $\mathcal{I}$ consists of an interpretation function I and a modal frame $F = \langle D, W, w_0, R \rangle$, where D is a domain, W is a set of worlds, $w_0 \in W$ is the designated actual world, and R is a binary relation on W (called* accessibility relation*). I is applied to two arguments: (1) a world and (2) a term, predicate symbol, or formula. For a given world w,*

- *to every term t, I assigns an object $I(w,t) \in D$*
- *to every n-ary predicate symbol P, I assigns a relation $I(w,P) \in D^n$*
- *for an atom $P(t_1, \ldots, t_n)$,*
  $I(w, P(t_1, \ldots, t_n) = true$ iff $\langle I(w,t_1), \ldots, I(w,t_n) \rangle \in I(w,P)$
- *for a complex non-modal formula ϕ, the truth value $I(w,\phi)$ recurs to the truth values of its sub-formulas as defined for predicate calculus (see Definition 3.6)*

[3] In case of reasoning about belief, the term "doxastic logic" is also used, while "epistemic logic" refers to reasoning about knowledge. We will not make this distinction here.

- $I(w, \Box\phi) = true$ *iff* $I(w', \phi) = true$ *for all worlds* w' *such that* $R(w, w')$

- $I(w, \Diamond\phi) = true$ *iff there is a world* w' *such that* $R(w, w')$ *and* $I(w', \phi) = true$

A formula ϕ *is true in a modal logic interpretation* $\mathcal{I} = \langle I, F \rangle$ *iff* $I(w_0, \phi) = true$*.*

In epistemic logic, the reading of modal formulas $\Box\phi$ is changed to "ϕ is believed" or "ϕ is known". Therefore, in work on reasoning about knowledge and belief, the modal operator $\Box$ is often replaced by special operators like B for belief or K for knowledge. Similarly, it is possible to replace $\Box$ by non-epistemic operators like W for "...is wanted" (or "...is a goal"). If there is more than one operator (i.e., more than one modality) in a logic, it is called *multi-modal*. Since all operators in a multi-modal logic are variants of the $\Box$ operator, they can be written as parameterized $\Box$ operators, using notations like $[B]$ or $\Box_B$. With such notations, the operators can also have $\Diamond$ variants, $\langle B \rangle$ or $\Diamond_B$.

Often it is necessary to reason about several agents; in this case separate operators with agent indices, like B_a, B_b etc., can be used. Logics that handle multiple agents are called *multi-agent logics*. If several operators and several agents are employed in a logic, it is a multi-modal multi-agent logic. Agents can be regarded as additional parameters of the standard operator $\Box$, so that B_a is sometimes also written $[B, a]$ or $\Box_{(B,a)}$. In this thesis, the latter notation will be employed.

In a multi-modal multi-agent logic each modality-agent combination induces a distinct modal operator. In order to distinguish the different operators semantically, a private accessibility relation $R_{(m,a)}$ is needed for all pairs of modalities m and agents a. In this case, a frame of a multi-modal logic can be defined as $\langle D, W, w_0, \mathcal{R} \rangle$, with $\mathcal{R}$ being a family of accessibility relations

$$\mathcal{R} := \{R_{(m,a)} \mid m \text{ is a modality, } a \text{ is an agent}\}$$

A similar effect could be achieved by introducing additional modality and agent parameters for R.

The truth of modal expressions depends on the accessibility relation R. Without any restriction on R, the following axioms hold in a modal logic with possible worlds semantics[4]:

1. all valid sentences in classical predicate logic (or propositional logic, respectively)

2. $\Box(\Phi \to \Psi) \to (\Box\Phi \to \Box\Psi)$ (K)

[4] Capital Greek letters denote formula variables.

Two inference rules can be used to deduce further valid sentences:

$$\frac{\Phi \quad \Phi \rightarrow \Psi}{\Psi} \quad \text{(modus ponens)}$$

$$\frac{\Phi}{\Box\Phi} \quad \text{(necessitation rule)}$$

It is easy to see that modus ponens and K can be summarized into the following formula, which we call *modal modus ponens*:

$$\Box\Phi \wedge \Box(\Phi \rightarrow \Psi) \rightarrow \Box\Psi \tag{3.2}$$

A single-operator logic with the above properties is a normal modal logic, also labeled K. In the following, multi-operator logics will be important. In a logic with multiple operators, different properties may hold for the different accessibility relations. Therefore, we will apply labels to the different modal operators. An operator with the above properties will be called K operator.

There is a severe problem immediately related to the above axioms, which in the literature on epistemic logic is called the problem of *logical omniscience*. (Modal) Modus ponens causes agents to believe (know, want, etc.) all logical consequences of their beliefs (knowledge, goals, etc.). The necessitation rule implies that agents believe (know, want, etc.) all tautologies. There has been a lot of research in the area of epistemic logic that tries to cope with logical omniscience. It is beyond the scope of this thesis to discuss the wide range of work on modal and epistemic logic in detail. For an overview, see, e.g., [Reichgelt, 1989; McArthur, 1988; Fagin *et al.*, 1995].

Different restrictions on an accessibility relation lead to different axiomatic characterizations of the corresponding modal operator and hence to different interpretations. The most common restrictions and their corresponding axioms are:

reflexive	$\Box\Phi \rightarrow \Phi$	(T)
serial	$\Box\Phi \rightarrow \neg\Box\neg\Phi$	(D)
transitive	$\Box\Phi \rightarrow \Box\Box\Phi$	(4, positive introspection)
euclidean	$\neg\Box\Phi \rightarrow \Box\neg\Box\Phi$	(5, negative introspection)

From a philosophical point of view, there seems to be a consensus that a KD45 operator, i.e. a K operator with additional properties D, 4, and 5, is adequate for reasoning about belief.

In general, work on user model representation and reasoning also deals with modeling beliefs and other attitudes and therefore is related to epistemic logic, particularly if logic-based mechanisms are employed. Therefore, such work is demanded to relate its approach to the above-mentioned standard characterizations of epistemic logic. In Chapters 5 and 6, reasoning methods will be developed that are related to modal logic. These methods implement modal operators as KD operators. The T axiom is not relevant to this work, because we presume that all knowledge in a UMKB is assumed by the system, so that there is no true belief. Introspective notions are not implemented, since in the area of user modeling a demand for reflexive assumptions has not been observed yet.

3.4.2 Modal Logic for User Modeling

Combining modal logic and plan recognition A first example for the application of modal logic in user modeling is a method for combining plan recognition with a model of the user's beliefs, knowledge, and interests, which is presented in [Ardissono and Sestero, 1996]. This work extends the plan recognition approach of Carberry [Carberry, 1989; Sarner and Carberry, 1992] by having the plan recognition algorithm interact with a modal logic user model. There are two main interactions: First, the plan recognition process accesses the user model in order to disambiguate its plan hypotheses. Second, from the current model of the plan that the user is about to perform (represented as a "context model", as proposed by Carberry), statements about the user's intentions are inferred that become stored in the user model.

Ardissono and Sestero's user model representation can be regarded as slightly confusing. Syntactically, it reminds of formalisms like those of [Moore and Paris, 1992] or [Kass, 1991], where first-order predicates were used to make statements about what the user believes or wants. Similarly, it is related to the formalism used in [Appelt and Pollack, 1992], where rules could be formulated to express relationships between different attitudes of the user. However, we will see that the formalism can indeed be considered a modal logic. Ardissono and Sestero offer a rich set of operators for expressing knowledge, belief, and intentions. They allow the representation of an interesting range of different assumption types. The notation of the operators is predicate-like, with agents and formulas as operator arguments:

- $\mathrm{Bel}(agt, p)$ for beliefs
- $\mathrm{K}(agt, p)$ for knowledge; it is defined as $\mathrm{Bel}(agt, p) \wedge p$
- $\mathrm{Goal}(agt, p)$ for goals
- $\mathrm{Know\_about}(agt, conc)$ is defined as $\mathrm{K}(agt, \mathrm{concept}(conc))$ (cf. Section 3.3.2)
- $\mathrm{Know\_if}(agt, p)$ is defined as $\mathrm{K}(agt, p) \wedge \mathrm{K}(agt, \neg p)$
- $\mathrm{Intend1}(agt, a)$ expresses the intention to perform an action, while
- $\mathrm{Intend2}(agt, p)$ expresses the intention to satisfy a condition

Two more operators can be used to express further specific kinds of knowledge.

In contrast to the above-mentioned approaches of using first-order predicates for the representation of user attitudes like belief, knowledge, and intention, here the p arguments may be complex first-order formulas, which are no terms of the language. So, we have a set of epistemic operators that are all variants of the modal $\Box$ operator. Ardissono and Sestero characterize the 'Bel' operator as a weak S5 (i.e., KD45) operator, which is the usual characterization of a belief operator. Some of the other operators are not that usual. Particularly the various knowledge operators are cumbersome; their applicability is quite restricted, and they are merely "syntactic sugar", i.e. abbreviations of specific expressions involving the main knowledge operator 'K'.

The user model is one knowledge base, the contents of which are represented with the above operators, the user being denoted by a specific agent symbol. All examples for user model contents that are presented by the authors are "modal literals", i.e. simple formulas of the form

$$[\neg]Op(agt, p)$$

where Op is one of the modal operators, agt is an agent symbol, and p is a first-order formula. Such a basic modal expression is very similar to the *type*:*content* notation of assumption type expressions. Together, Op and agt determine the type of an assumption, and p can be regarded as assumption content. This correspondence between simple modal formulas and assumption type expressions will be exploited in Chapter 5 to build a bridge between assumption types and modal logic.

Possible content expressions concern either conceptual knowledge or planning knowledge. Above, we already mentioned the 'concept' predicate which has the usual meaning. All other conceptual knowledge is expressed in a first-order notation, e.g.

$$\text{Bel}(U, \forall x \text{ Professor}(x) \rightarrow \text{Department-employee}(x))$$

represents the assumption that the user believes that the concept 'Professor' is a subconcept of 'Department-employee'.

Planning knowledge consists of action decomposition and descriptions of the effects and conditions of actions. Hence, among the representation primitives for planning knowledge are the first-order predicates

decomp(act, d)	d is a decomposition of action act
in-decomp(act, d)	action act is part of a decomposition d
post(act, x)	x is a postcondition or effect of action act

Note that at first sight, two separate kinds of assumption contents are employed, namely conceptual knowledge and planning knowledge. However, FOPC is used as content formalism for both kinds of knowledge. This makes sense because conceptual and planning knowledge are not completely independent of each other. Individual assertions of conceptual knowledge like 'Professor(Sauerbruch)' can be employed for formulating effects or conditions in planning knowledge.

Furthermore, complex formulas of the modal language are used to express acquisition rules. These rules are applied to infer additional assumptions from existing user model contents. Since different operators can appear in acquisition rules, the rules establish relationships between assumptions of different types. An example is the complex rule

$$\begin{aligned}&[\forall d\, \mathrm{Bel}(agt, \mathrm{decomp}(act, d)) \rightarrow \\ &\quad \exists a\, \mathrm{Bel}(agt, \text{in-decomp}(a, d)) \wedge \neg \mathrm{Intend1}(agt, a)] \\ &\rightarrow \neg \mathrm{Intend1}(agt, act)\end{aligned}$$

It says that an agent does not want to perform an action *act* if she believes that there is a component action *a* she does not intend to do.

Also stereotypes are represented, namely as a set of modal formulas that represent assumptions about the user by having the agent symbol U as first argument. Stereotype activation is controlled by trigger formulas referring to the current user model. The stereotype knowledge bases can be ordered in a hierarchy.

Hustadt's modal description logic In Section 3.3, it was shown that concept-based formalisms are fairly often employed in user modeling systems. Currently, description logics are the most widely acknowledged concept-based representation formalisms. In [Hustadt, 1994; Hustadt, 1995], Hustadt extends the description logic $\mathcal{ALC}$ with the generic modal operators $\Box_{(m,a)}$ and $\Diamond_{(m,a)}$. Thus, he develops a multi-modal, multi-agent description logic, which allows to formulate so-called "subjective" concepts and roles by applying modal operators to concepts and roles. Also, terminological and assertional sentences can be made relative to agents' attitudes. In principle, Hustadt's formalism, called $\mathcal{FALCM}$, allows general agent modeling even in situations of group communication. Because it introduces a powerful approach to representing and reasoning with stereotyped assumptions, $\mathcal{FALCM}$ also is very interesting for the more specific user modeling scenario, with system and user being the interacting agents.

Imagine a system modeling the user's assessment of cars[5]. An example for a subjective concept is

$$\Box_{(B,U)}\ \text{nice-car}$$

It denotes the class of all objects that are believed by the user to be nice cars. Similarly, a modal operator can be used to represent terminological expressions

[5] The examples in [Hustadt, 1995] are also from the car domain.

and individual assertions relative to a modality and an agent:

$$\Box_{(B,U)}\ \text{bmw} \sqsubseteq \text{nice-car}$$

states that the user believes that BMWs are nice cars, and

$$\Box_{(B,S)}\Box_{(B,U)}\ \text{BMW1} \in \text{bmw}$$

says that the individual car 'BMW1' is a BMW, according to what the system assumes about the user's beliefs.

The most interesting property of Hustadt's formalism is that agent symbols are normal object symbols of the terminological language, and that concepts therefore can describe classes of agents. Such agent concepts A are used in combination with the special modal operators $\Box^c_{(m,A)}$ and $\Diamond^c_{(m,A)}$ for common (i.e., mutually shared) attitudes of an agent class, and $\Box^g_{(m,A)}$ and $\Diamond^g_{(m,A)}$ for group attitudes of an agent class. These operators can be employed to formulate stereotyped assumptions, which can be held by any agent. For example,

$$\Box_{(B,S)}\Box^g_{(B,\text{porsche-owner})}\ \text{bmw} \sqsubseteq \text{slow-car}$$

means that the system believes all Porsche owners to classify BMWs as slow cars.

The canonical triggering condition for the 'Porsche owner' stereotype is

$$\Box_{(B,S)}\ \exists\text{own.porsche} \sqsubseteq \text{porsche-owner}$$

which simply says that the system believes that everyone who owns a Porsche is a Porsche owner. If, in addition,

$$\Box_{(B,S)}\ \text{Porsche4} \in \text{porsche}$$

(i.e., the system knows about the 'porsche' instance 'Porsche4') and

$$\Box_{(B,S)}\ (U, \text{Porsche4}) \in \text{own}$$

(stating that the user owns the 'Porsche4' according to system beliefs) hold, it can be deduced that

$$\Box_{(B,S)}\ U \in \text{porsche-owner}$$

With such assertions about agent objects belonging to an agent class, stereotyped assumptions can be assigned to an agent. Since assertions can be made within the scope of modal operators, such stereotype assignments are relative to an agent. By the current classification of the user as Porsche owner, the belief about BMWs being slow cars which the system stereotypically ascribes to Porsche owners (s.a.) is assigned to the user, so that

$$\Box_{(B,S)}\Box_{(B,U)}\ \text{BMW1} \in \text{slow-car}$$

can be inferred.

In the above examples, assumptions of the types SB and SBUB and stereotyped assumptions were formulated only. This may lead to a false impression. Hustadt's formalism is much more powerful. The following is a slight modification of an example given in [Hustadt, 1995]:

$$\Box_{(B,a)}\,(\text{nice-car} \sqsubseteq \Box^{g}_{(B,\text{porsche-owner})}\text{slow-car})$$

This expression means that agent a believes that nice cars are among the objects that are believed to be slow cars by Porsche owners. Obviously, such complex expressions cannot be split into assumption type and content.

To implement reasoning procedures for $\mathcal{FALCM}$, Hustadt makes use of the translation approach to modal logic theorem proving. In this approach, algorithms are used that translate modal formulas and their semantics into first-order logic, such that first-order theorem proving or reasoning techniques can be utilized. Hustadt employs translation techniques that are based on ideas of [Moore, 1980] and [Nonnengart, 1992]. In addition, the algorithm SCAN [Gabbay and Ohlbach, 1992] is used that translates modal axioms to FOPC. SCAN allows to define the properties of the modal operators of $\mathcal{FALCM}$ by adding the necessary modal axioms to a knowledge base. Their SCAN translations are processed together with the translated knowledge base using first-order mechanisms.

The AsTRa framework for user model representation, which will be presented in Chapter 5, integrates modal logic representation and reasoning mechanisms with simpler assumption type representation. In Chapter 6, it will be described how the user modeling shell system BGP-MS implements the AsTRa framework. In this implementation, translation techniques similar to those of Hustadt have been used for realizing modal logic reasoning. The translation approach will therefore be discussed in more detail in Section 6.4.1.

3.5 Summary

We have described a large number of user modeling systems in this chapter that represent their user modeling knowledge either in a logic-based way or in a way that is closely related to logic-based mechanisms. For all systems, a characterization of user model contents in terms of assumption types and assumption contents was attempted. The overview given in this chapter is summarized in two tables, one covering pure assumption content representation (Table 3.2), the other listing work on representing assumption type and content (Table 3.3). Both tables list representation formalisms, systems described in the literature that use these formalisms or representations equivalent to these formalisms[6], and the involved assumption types. In the assumption type labels, Ach stands for "achieves", C

[6] The table does not exhaustively list all systems mentioned in this chapter.

Table 3.2: Representation in user modeling: assumption content only

formalism	*remark*	*system/literature*	*ass. types*
propos. calculus	atoms	[Paris, 1989]	SBUB
		[Nwana, 1991], [Tattersall, 1992], [Shifroni and Shanon, 1992] TECHDOC-I [Peter and Rösner, 1994]	SBUB*
		UMFE [Sleeman, 1985]	SB*UB
		Grundy [Rich, 1983]	SB*UPref*
	rules	SMMS [Huang *et al.*, 1991]	SBUB, SB∼UB
FOPC	atoms	[Wu, 1991]	SBUW(UB)
		REPLAI-II [Retz-Schmidt, 1991]	SB*UPref
	variables	TRACK [Carberry, 1989]	SBUW
	rules	ANATOM-TUTOR [Beaumont, 1994]	SBUB
		[Quilici, 1989; Quilici, 1994] GUMAC [Kass, 1991]	SBUB, SB
special	frames	CHAMP [Lehman and Carbonell, 1989], TECHDOC-I [Peter and Rösner, 1994]	SB
	concepts	UMFE [Sleeman, 1985], [Moore and Paris, 1992], GUMAC [Kass, 1991]	SB, SBUB
	others	TRACK [Carberry, 1989], [Goodman and Litman, 1992], [Niem *et al.*, 1993]	div.

Table 3.3: Representation in user modeling: assumption type and content

formalism	*system/literature*	*assumption types*
FOPC	[Moore and Paris, 1992]	SBUB, SBUW, SBUC
	[Quilici, 1989; Quilici, 1994]	SBUB, SB
	[Appelt and Pollack, 1992]	SBUB, SBUInt, SBUAch, SBUExp
modal logic	[Hustadt, 1994; Hustadt, 1995]	SBUB, ...
	[Ardissono and Sestero, 1996]	SBUB, SBUK, SBUW, SBUInt

for "has competence", Exp for "Expects", Int for "intends", K for "knows", and Pref for "prefers". The contents of the remark column in Table 3.2 indicate what features of the corresponding formalism are used.

The following lessons have been learned from the analysis of the described systems:

- Some systems use assumptions of one type only, which are stored in one knowledge base. This is the most simple approach to user model representation and the most basic way of following the idea of assumption type representation. .

- Several systems employ a domain knowledge base (i.e., SB knowledge) and express assumptions about the user by attaching graduation values to domain knowledge items, e.g. [Sleeman, 1985; Shifroni and Shanon, 1992; Tattersall, 1992; Peter and Rösner, 1994]. This fairly popular *overlay* approach does not directly fit the idea of separate assumption type representation, in particular if assumption content representation is logic-based and does not permit graduations. However, an overlay model of beliefs can be approximated by maintaining SB and SBUB assumption types, with SBUB assumption contents being a subset of SB contents. If it were possible to attach graduation values to assumption contents (of SBUB), the typical SBUB* assumptions of overlay models could be represented. In a purely logic-based approach, a three-level graduation can be established by additionally maintaining an SB$\sim$UB assumption type. Thus, a representation like that of UMFE (cf. Section 3.1.2) can be realized with assumption types.

- A third possibility is to maintain assumptions of different types within one knowledge base. Then, assumption type information is expressed employing the syntactical facilities of the (logical) representation formalism. Most systems that follow this approach utilize a more or less powerful subset of first-order predicate calculus together with special assumption type predicates like 'Bel', 'Goal', etc. [Moore and Paris, 1992; Kass, 1991; Quilici, 1989; Quilici, 1994; Appelt and Pollack, 1992]. A similar method is to employ a modal logic with epistemic operators that represent assumption type information [Hustadt, 1995; Ardissono and Sestero, 1996]. Modal logics are more expressive, particularly as far as assumption contents are concerned. It has been shown that in most of these systems, the genuine assumptions about the user can be translated into an assumption type representation. However, the complexity of rules that serve for user model acquisition purposes goes beyond assumption types, since these rules mostly involve assumptions of different types. So, it would be beneficial to extend an assumption type representation with possibilities for establishing relationships, in particular inference relationships, between assumption types. Maximal expressive power can be achieved, if such an extension is realized with modal logic.

- In the modal logic representations of both [Hustadt, 1995] and [Ardissono and Sestero, 1996], the behavior of modal operators is characterized with axiom schemes. For example, all knowledge operators of [Ardissono and Sestero, 1996] are characterized by axioms that are based on the belief operator. This operator itself is characterized by classical axioms of modal logic. In an assumption type representation with a modal logic extension, it would be nice to have the capability of processing modal axioms. Then it would be possible to freely introduce operators and define their semantics using axiom schemes. In addition, generic inference rules involving several assumption types can be expressed with schematic modal formulas.

- Assumptions about the user may concern quite different entities of domains with quite different properties. Therefore, specialized representation formalisms (relative to general logical formalisms) have frequently been employed for user modeling. Specialized formalisms are mostly used for assumption contents only. Particularly concept-based formalisms, but also special formalisms for planning knowledge are used. In some systems, several kinds of assumption contents are used, even within the same assumption type. Examples are GUMAC [Kass, 1991] and the plan recognition system of [Ardissono and Sestero, 1996]. In these systems, beliefs of the user concerning domain concepts and plans are represented. While Ardissono and Sestero represent both kinds of contents with predicate calculus, Kass implements concept and plan knowledge with different specialized formalisms. In general, the availability of different formalisms for assumption contents allows to use the most appropriate representation for different kinds of contents.

The AsTRa framework for assumption-type representation, which will be presented in Chapter 5, is based on the idea of separating user model contents of different assumption types into distinct partial knowledge bases, following the partition approach. Furthermore, it is designed to allow the integration of several formalisms for assumption content representation. In the AsTRa framework, user modeling knowledge bases can be described in terms of modal logic, so that it is possible to integrate modal logic representation and reasoning techniques into the framework.

Chapter 4

Reasoning in User Modeling Systems

One of the most important properties of a logic-based knowledge representation formalism is its capability to draw conclusions from explicitly represented knowledge, thus permitting access to the knowledge that is only implicitly contained in a knowledge base. This capability is useful also for user modeling purposes. In most user modeling applications, it is difficult to directly acquire assumptions about the user that are sufficient to support intended individualizations. Normally, users will not tell the system voluntarily and explicitly about their individual properties and attitudes. Furthermore, it is generally not recommended to pose many direct queries to the user, in order to avoid intrusiveness. So, it becomes necessary to reason about the user, i.e., to infer the needed assumptions about her.

There are many different ways to categorize reasoning methods or inference mechanisms. As far as user modeling systems and, in particular, user model acquisition are concerned, the following two types of inferences can be distinguished:

- Inferences can be performed outside the user model. For instance, user model contents can be inferred from the user's behavior, as far as it can be observed by the system. A dialog history can be considered so that the inference process does not have to rely on a single, current observation only. Another possibility is to conclude assumptions about the user from system behavior. For instance, in a natural language explanation system, the user may be assumed to know a concept or at least to know it better than before after the system explained it to her.

  If successful, such inferences cause new assumptions to be entered into the user model, which are not necessarily related to other user model contents. Such assumptions which stem from outside the user model are called *primary assumptions*; we will refer to the inference processes that lead to primary assumptions as *primary inferences*.

- Inferences can also be performed inside the user model. That is, new assumptions about the user can be derived from existing user model contents. For instance, it can be plausible to assume that a user knows a fact if she is already assumed to know many related items.

  Such inferences lead to assumptions about the user that are dependent of other user model contents. They come from within the user model. In contrast to primary assumptions they are called *secondary assumptions*. We will also speak about *secondary inferences*.

Inference mechanisms of knowledge representation formalisms are more appropriate for secondary inferences than for primary inferences. They draw conclusions from explicitly formulated knowledge items and therefore are designed to operate "within" a knowledge base. However, it is imaginable to apply such an inference engine also for primary inferences. As a prerequisite, user observations or statements about system behavior would have to be coded in the respective representation formalism.

In this chapter, we will present several examples of inference methods that were employed in user modeling systems. In the user modeling literature, it is often the case that acquisition or inference methods are mentioned but not described in detail, let alone a formal description. Only such user modeling work that presents inference methods in sufficient detail was chosen for this review. Our focus is on logic-based reasoning and inferences, but also related methods will be described in order to identify standard reasoning tasks that are often performed by user modeling systems. Implementations of such tasks should be provided by a user modeling shell system.

4.1 Forward-Directed Reasoning for User Model Acquisition

The classical way of doing logic-based, deductive reasoning is to derive a conclusion from given premises. Formal deductive systems, from Aristotelian inference patterns to Hilbert calculi, mainly consist of inference rules that permit to form sentences that follow from previously given sentences. Therefore, it seems natural to apply logic-based reasoning mechanisms in knowledge-based systems in a *forward-directed* manner, i.e., to start with a given knowledge base and/or a given expression and determine logical consequences. In this section, we present several applications of forward-directed reasoning in user modeling for the acquisition of both primary and secondary assumptions. Typically, forward inferences are invoked when a new observation about the user has been made or when a new assumption about the user has been acquired, respectively. The new observation or assumption then constitutes the starting point for the forward-directed reasoning process.

4.1.1 Primary Acquisition in KNOME

KNOME, the user modeling component of the "Unix Consultant" UC, was developed by Chin and is described in [Chin, 1986; Chin, 1989]. Both of the two main contributions of KNOME deal with user model acquisition: The double-stereotype approach, which will be discussed in Section 4.3, and the utilization of a set of rules for primary inferences. Three of these rules are presented here, which show what kinds of information can be used to infer assumptions about the user's knowledge in KNOME [Chin, 1989, p.86]. All rule premises (the left hand sides of the implications) refer to observed user actions or utterances, not to user model contents. Hence, the rules enable primary inferences.

- user states that user does (not) know X → user does (not) know X

  The condition part refers to a user statement. KNOME will get corresponding inputs from the natural-language analysis component of UC.

- user wants to know X → user does not know X

  This rule refers to the current (information) goal of the user as uttered by her.

- user uses X → user knows X

  This rule refers to a user action. It is specific to domains where items that can be used can also be characterized as (not) known to the user. In case of KNOME, such items are UNIX commands.

All rules that Chin presents are domain-independent. Only the last rule requires a certain class of domains in order to be applicable.

4.1.2 Secondary Acquisition with Modal Logic

The plan recognition system that is described in [Ardissono and Sestero, 1996] employs modal logic expressions as user model contents. A rich set of modal operators is utilized to represent assumptions about the user's knowledge, beliefs, goals, and intentions. Besides, predicate logical primitives are used to describe planning knowledge. In Section 3.4.2, the formalism of Ardissono and Sestero was described. Recall that 'Bel' is a belief operator, 'Goal' is a goal operator, 'Intend1' means the conscious intention to perform an action, and 'post' is used to denote the effects (the postconditions) of an action. These means are needed for formulating one of the four acquisition rules that are an original contribution of Ardissono and Sestero:

$$\mathrm{Bel}(agt, \mathrm{post}(act, c)) \wedge \mathrm{Goal}(agt, \neg c) \;\rightarrow\; \neg\mathrm{Intend1}(agt, act)$$

This is the original notation; non-complex terms (in italics) are to be interpreted as universally quantified variables. The rule says that an agent does not want to perform an action if she believes that the action has an undesired effect. Like the other presented rules, it supports secondary inferences, since the rule premises refer to assumed beliefs of an agent, which are already part of the agent model.

Together with the other rules, the above formula is applied in a forward manner to "expand" the user model, as the authors put it. The main focus of the system is to infer assumptions about what is intended and what is not intended by the user, i.e. formulas of the form '$\mathrm{Intend1}(\mathrm{U}, act)$' and '$\neg\mathrm{Intend1}(\mathrm{U}, act)$'. In addition to forward inferences based on acquisition rules, a specialized procedure is employed that constructs such expressions from the results of the plan recognition process.

4.1.3 UMFE: Inferences Based on Domain Knowledge

The system UMFE [Sleeman, 1985], the "User Modeling Front-End" of the expert system NEOMYCIN, employs the domain knowledge of NEOMYCIN for extending its user model with secondary inferences. In the domain knowledge base, a taxonomy of bacterial infections is represented by a concept hierarchy. Each concept is labeled with two numerical values that characterize the difficulty and the importance of the concept with respect to second-year medical students.

UMFE uses two general rules for secondary inferences. However, these rules do not only refer to existing user model contents, but also exploit the hierarchical relationships between domain concepts as well as their importance values. The first general inference rule of UMFE is:

> If a concept is known / not known, then so will its parent concept in the hierarchy and all of its siblings which have comparable or higher importance values.

This rule is applied recursively until a pre-defined limit is reached. A second inference rule is only applied to primarily acquired assumptions, i.e. it considers only such concepts that the user has explicitly stated to know or not know. This second rule (which, more precisely, can be split into two positive / negative variants) is:

> If the user has said to know / not know a concept, she will be assumed to know / not know concepts with lesser / greater difficulty.

Possible conflicts between these rules are resolved with the help of a precedence relation, which rates the user to be the most reliable source of information and the second rule to be least reliable. More reliable sources are preferred.

Similar to UMFE, the adaptive hypertext system KN-AHS [Kobsa *et al.*, 1994] bases its secondary inferences about the user's knowledge of domain concepts on terminological domain knowledge. In case of KN-AHS, this knowledge is available in the user modeling knowledge base.

4.1.4 Acquisition Rules in GUMAC

The system GUMAC, which was developed by Kass, mainly contributed to the exploration of domain-independent methods for user model acquisition. In [Kass, 1991], thirteen rules are presented that can be used to infer assumptions about the user mainly from existing contents of a user modeling knowledge base, including system knowledge, but also from observed behavior. So, most rules support secondary inferences, but some of them are also of a primary nature. In Section 3.2.3, two rules were presented as examples for the potential of logical formalisms in user modeling. Kass describes all acquisition rules as default rules using the symbol $\rightsquigarrow$ for default implication. For both implication premises and conclusions, a purely logical language is used. However, there is no logical inference engine for processing these rules, so that the rule formalism is only of descriptive use. Kass notes that "the user model acquisition rules are implemented as condition-action pairs, where the condition is a pattern that can match elements of the domain knowledge base, the user model, or an MRL expression" [Kass, 1991, p.222]. MRL is the "meaning representation language", an output language of a natural-language understanding system that is supposed to work as an interface system to GUMAC. An MRL expression contains information about speech acts the user has made, i.e., information about observed user behavior. The different acquisition rules access this information by including premise expressions of the form

$$\mathrm{observe}(A, \mathrm{do}(A', \mathit{Action}))$$

or, more simply,

$$\mathrm{do}(A, \mathit{Action})$$

where A and A' are agents (the system or the user), and *Action* is a speech act. The other language elements that occur in rule descriptions are the representational primitives for definitional (i.e., terminological) and strategical knowledge, a '$\mathrm{Bel}(A, p)$' predicate for stating that an agent A believes an expression p, and simple logical expressions for referring to domain knowledge. Most of these facilities have been presented in Section 3.2.3.

An example of a purely primary acquisition rule is the "observed action rule" (presented here in a slightly modified notation)

$$\mathrm{observe}(A, \mathrm{do}(A', \mathit{Action})) \rightsquigarrow \mathrm{Bel}(A, \mathrm{achieved}(A', \mathit{Action}))$$

saying that if an agent observes an action, then it will believe the action has been performed. If A is the system and A' is the user, then the conclusion of the above rule may cause other rules to fire that have

$$\mathrm{achieved}(U, \mathit{Action}) \rightsquigarrow \ldots$$

as condition part[1]. It is a bit difficult to characterize such rules precisely. The

[1]This condition matches the conclusion of the "observed action rule", since system beliefs can be expressed without the 'Bel' predicate.

crucial point is, whether or not an 'achieved($U, Action$)' expression is regarded as part of the user model or not. If so, the rules are secondary inference rules, because they rely on already inferred knowledge about the user. If not, i.e., if an 'achieved' expression is regarded as system knowledge about observed user behavior that immediately results from a simple transformation of an 'observe' expression, these rules are primary rules.

Rules that are based on observed actions but mix contents of the user model into the conditions can be regarded as rules that combine primary and secondary inferences. An example is the "expert rule" (cf. Section 3.2.3)

$$\mathrm{Bel}(U, \mathrm{expert}(A)) \wedge \mathrm{do}(A, \mathrm{tell}(A, U, p)) \rightsquigarrow \mathrm{Bel}(U, p)$$

that lets the user believe everything an expert tells her.

Purely secondary rules are all rules which are assumed to be held by the agent being modeled, the "local rules" in Kass's words. Local rules describe inferences that are drawn within the beliefs of one agent, i.e., within assumptions of one type. In GUMAC, local rules merely describe the built-in transitivity and inheritance inferences of the KL-ONE-like representation system NIKL, which is employed for representing hierarchies of domain concepts. An example is the "concept transitivity rule" (cf. Section 3.2.3)

$$\mathrm{isa}(A, B) \wedge \mathrm{isa}(B, C) \rightsquigarrow \mathrm{isa}(A, C)$$

which describes the transitivity of the IS-A relation in NIKL.

Local rules are rules of the modeled agent. So, if an inference is made based on a local rule, it reflects an inference the modeled agent may make and hence is of a simulative nature. In contrast, a rule like the "expert rule" does not support simulative inferences, but describes a rule that the system (not the agent 'S', but the user modeling system, in this case GUMAC) has at hand for meta-level user model acquisition. Similar meta-level rules are used by [Ardissono and Sestero, 1996] for secondary user model acquisition (cf. Section 4.1.2). So, another classification of inferences can be established:

- *simulative inferences* are inferences that a modeled agent (the user in most cases) is assumed to be capable of. That is, the agent must be assumed to hold an inference rule on its own and to be able to process it like the system does. Simulative inferences are likely to concern only one type of assumptions.

- *meta-level inferences* are inferences that a user modeling system can draw according to agent-independent rules. Such rules are typically pre-defined by the developer of a user modeling system. They can be general, like in GUMAC, but are likely to be domain-specific. Meta-level inferences may involve assumptions of different types.

Secondary inferences can be both simulative and meta-level inferences, while primary inferences are meta-level inferences in most cases. However, if system activities are also regarded as observations that can be exploited for primary inferences, it might be interesting to determine what the user might infer from such observations. There are systems that take user inferences from observations about system behavior into account. However, in these cases it is sometimes hard to decide whether a system performs simulative inferences (based on inference rules that are ascribed to the user) or meta-level inferences (based on rules of the system that make statements about what the user might conclude from system behavior). An example will be presented in Section 4.1.6.

GUMAC processes its acquisition rules in a forward chaining manner. The inference process is driven by the statements that are made in the user-system dialog. Starting with observed statements, GUMAC tries to apply as many rules as possible. This is a standard recursive forward chaining process; knowledge inferred with one rule can help to satisfy the conditions of other rules. Kass states that a probably better algorithm would be to draw only a limited number of inferences immediately by forward reasoning, and to further extend the user model by backward reasoning when information is requested from the user model.

The default character of the acquisition rules of GUMAC is supported by an ATMS-like truth maintenance system. In GUMAC, an assumption about the user is not retracted, even if it is inconsistent with another assumption, as long as there is any justification for it. Thus, GUMAC is able to model inconsistent beliefs of a user.

4.1.5 Propositional Reasoning in SMMS

In [Huang *et al.*, 1991], the system SMMS (Student Model Maintenance System) is presented. The main point of SMMS is the application of truth maintenance techniques in student modeling. SMMS provides a domain-independent method for the combined revision of deductive and stereotyped knowledge. Deductive knowledge is the user's knowledge about concepts of the tutoring domain. The user model may contain positive or negative literals of propositional calculus, which represent the concepts that the user is assumed to know or not know, respectively (cf. section 3.1.3). In addition to concept literals, logical implications are available in SMMS. They are of the form

$$A_1 \wedge \ldots \wedge A_n \rightarrow [\neg]B$$

That is, these rules can be used to infer a positive or negative literal B from a number of atoms A. In SMMS, rules are processed by a propositional deduction engine in a forward manner to infer new statements about the user's conceptual knowledge.

The deductive rules of SMMS are system rules for meta-level inferences, meta-level!inferences although they look like internal rules of the formalism that is employed for representing assumptions about the user. Since only one assumption type is available in SMMS, it is not necessary to add assumption type information to the literals of the rules. Furthermore, SMMS inferences can be characterized as secondary inferences, because the left hand side of the implication can be satisfied by user model contents only. Acquisition of primary assumptions is done by the subcomponent SKAS, the "student knowledge analysing system". Conflicts that may occur in the user model as a consequence of the forward inference process are resolved by applying the truth maintenance techniques that were developed for SMMS.

4.1.6 Considering User Inferences for Explanations

The work of Zukerman is concerned with user-adaptive generation of natural-language utterances. In [Zukerman and McConachy, 1993], a method for planning the content of utterances in a tutoring environment is presented, which considers inferences the user (the student) may draw from a system utterance. Like other approaches that are mentioned by Zukerman and McConachy, their system considers mainly two phenomena: First, it takes into account that a user might draw erroneous conclusions from the propositions conveyed in a system utterance and adds correcting propositions if necessary. Second, it checks which propositions the user can easily infer from a system utterance and omits such propositions. For these purposes, the system has a set of inference rules that it can apply to simulate possible user inferences. The premises of these rules do not only depend on the system utterance, but also on the current user model contents.

In the user model, assumptions about the beliefs and skills of the user are represented. The representation formalism covers a class of technical domains like algebra, where procedures exist that can be applied to particular objects to achieve certain goals. The user is modeled to have beliefs about the relations between procedures, goals, and objects (e.g., 'has-goal' and 'apply-to') and to have skills with respect to procedures. Assumptions about user beliefs are graduated; the strengths of system conjectures about user beliefs are represented with the help of six qualitative values like 'believed', 'rather disbelieved', etc., so that SB*UB assumptions are modeled[2]. Consequently, also the inference rules involve computations of uncertainty values. The general form of an inference rule is

$$\textit{Rule-name(RD[, beliefs])} \xrightarrow{L_{rule}, C_{rule}} \textit{Belief} \text{ and/or } \textit{Skill}$$

RD is short for "rhetorical device", which is an utterance pattern. An utterance may be the assertion or the negation of a proposition, so that RDs can have the form 'Assert(p)' or 'Negate(p)'. Then, a rule infers a *Belief* and/or a *Skill* from

[2]Zukerman and McConachy state these values to represent the "strength of a student's conjectured belief" [Zukerman and McConachy, 1993, p.161]. However, they do not graduate SBUB* assumptions, as it might seem, but system assessment of user beliefs, i.e., SB*UB assumptions.

an utterance, possibly constrained by beliefs that are already held by a student. L_{rule} is the likelihood that the user will conclude that an asserted proposition is true or a negated proposition is false. The strength of the user's belief in a proposition is determined by the confidence value C_{rule} and the strength values of the *beliefs*. L_{rule} and C_{rule} values determine the abilities of a student. Zukerman and McConachy employ 5 sets of L/C values for all rules, which they call "student profiles", to further individualize inference processes.

There are three types of rules, each of which can involve an assertion or a negation. The 'Generalize' rule for an assertion (briefly labeled $G+$) is

$$\text{Generalize}(\text{Assert}(P(Z)), \text{inst}(Z, Z')) \overset{L_{G+}, C_{G+}}{\longrightarrow} \text{Belief}(P(Z'))$$

This rule states that, given an assertion of the proposition $P(Z)$ and the user belief that Z is an instance of Z', the user will conclude with likelihood L_{G+} that $P(Z')$ is true. The strength of the user's belief in $P(Z')$ is a function of C_{G+} and the strength of her belief in $\text{inst}(Z, Z')$.

It is not obvious how such rules shall be characterized. On the one hand, the authors state that the rules represent *the user's* possible inferences. The C_{rule} value is said to represent the confidence a user has in that rule. Through the employment of individualized L/C values, the impression is supported that inferences based on the rules are simulative. On the other hand, the rules are not assumed to be believed by the user. They are grounded on theories of natural-language understanding and can be employed by the system to draw inferences about the user. In this interpretation, the rules support meta-level inferences. The differing L/C values are then part of a user model that influence the application of the rules similar to the *beliefs* condition parameter.

The inference rules are applied in a forward chaining manner, with the currently planned assertion or negation as input. So, the system can check, which conclusions the user will draw from a system utterance. If she concludes something wrong, a correcting proposition can be added to the utterance. If a proposition that is planned to be in the utterance can be concluded, then it can be omitted. Furthermore, also backward chaining is employed to determine possible RDs for a proposition which shall become a belief of the student.

4.2 Backward-Directed Reasoning for Answering Queries to the User Model

The main purpose of a user model is that it can be accessed if information about the user is needed. So, a user modeling system must provide a mechanism for querying the user model. When an application system intends to individualize an aspect of its behavior, it can use the query mechanism to check if certain assumptions about the user are held or not, or, in case of graduated assumptions, to which degree they are held. A central property of logic-based knowledge representation systems is that a query to the knowledge base has access also to implicitly represented knowledge. For this purpose, the representation system employs an inference engine to check whether a queried item can be derived from the knowledge base. Such query-driven reasoning works in a backward-chaining manner, attempting to find knowledge base contents that entail the query. Also in logic-based user modeling systems, *backward-directed* reasoning can be used for answering queries.

However, backward inferences are not very often reported to be employed in user modeling systems. In the previous section, we cited a statement of Kass on this topic. His system GUMAC employs forward inferences, but Kass said that a better way for doing inferences would be to combine limited forward inferences, which are drawn at the time an observation is made, with backward inferences at the time of a query. This would be an improvement, because backward inferences are goal-directed and more focused, while (exhaustive) forward reasoning may lead to many new assumptions about the user that will never be needed by an application [Kass, 1991].

A slightly different kind of backward reasoning is employed by [Zukerman and McConachy, 1993] (cf. Section 4.1.6). In their system, inference rules are utilized for content planning in a tutoring environment. The rules describe the effects of system utterances on the beliefs of students. So, if the system intends to make the user believe in a certain proposition about the learning domain, it applies the rules in a backward-chaining manner to determine the utterances from which the intended belief can be inferred by the student. This process can be regarded as a "look-ahead query" to the user model.

In several papers, Quilici developed methods for natural-language advisory systems. In work on handling plan-oriented user misconceptions [Quilici, 1989; Quilici, 1994], he utilizes backward-directed deductive reasoning for verifying advisor beliefs about relationships between possible user actions and their effects. The representation formalism that is used in Quilici's approach was described in Sections 3.2.3 and 3.2.4. Deductive inferences can be drawn based on *explanation rules*, which are 'IF ... THEN ...' rules. Explanation rules express structural relationships within planning knowledge, or determine how an event is evaluated, possibly in relation to a goal that is to be achieved. Explanation rules can be used for inferences concerning both the system (the "advisor" in Quilici's terms) and the user. However, in the main algorithm of the advisory system [Quilici, 1989], only system beliefs are verified: The system processes user beliefs that result from a transformation of the user's natural-language problem description. First, the advisor determines if it holds the user belief itself. If this is not the case, the advisor attempts to verify beliefs that contradict the user belief, according to its planning knowledge. For both steps, backward reasoning based on the explanation rules can be applied. In [Quilici, 1994], explanation chains for both user and advisor beliefs are constructed based on simulative inferences. For this purpose, explanation rules are ascribed to the simulated agent; i.e., these rules are considered to be equally held by both advisor and user. Explanation chains for erroneous user beliefs will contain wrong assumptions of the user that need to be corrected by the advisor.

A similar, explanatory application of backward reasoning is the weighted abduction approach to plan recognition of [Appelt and Pollack, 1992], which was described in Section 3.2.4. In this approach, plan ascription rules describe inferential relations between the user's actions, beliefs and intentions. They are transformed into logical implications. When a user action is observed, a modified backward proof procedure is used to find a set of assumptions (that are needed to prove subgoals in the inference but are not proved themselves) which explains the observation. In contrast to the explanation rules of Quilici, the plan ascription rules are meta-level rules. This is not surprising, since plan recognition is a meta-level reasoning process by definition.

Although backward reasoning is rarely used for answering queries to the user model, it can be beneficial. We agree with Kass that backward reasoning is more goal-directed and therefore more efficient than forward reasoning. Since the main service of a logic-based representation formalism is to provide backward deductive reasoning procedures, a logic-based user modeling system will naturally offer the possibility to employ backward inferences for query answering. However, a combination of backward and possibly limited forward inferences seems beneficial. Kass' argument for a combination of both strategies is that a limitation of forward inferences will avoid too many superfluous entries into the user model. Backward reasoning should be employed to complement forward acquisition, so that no interesting assumptions about the user are missed.

An argument in favor of forward reasoning can be expressed in terms of Chin, who establishes a classification system for user model acquisition techniques [Chin, 1993]. On one dimension of this system, he distinguishes between *on-line* and *off-line* acquisition. On-line acquisition is meant to be done while the user is in the process of interacting with the adaptive program. Therefore, on-line acquisition ought to be limited with respect to its consumption of computing resources, particularly concerning its response time. Off-line acquisition is less problematic, since it is done independent of the user-system interaction, perhaps even after the interaction is finished. Answering queries with backward inferences is a typical on-line technique, because the application has to wait for the answer to its query. Also other uses of backward reasoning, like for content planning [Zukerman and McConachy, 1993] or for verifying system beliefs for an explanation [Quilici, 1989], require on-line inferences. In some cases, also forward inferences may have to be executed on-line, for example for assessing possible inferences that a user may draw from an intended system utterance [Zukerman and McConachy, 1993]. But the execution of forward inferences for secondary user model acquisition, like in GUMAC for instance, can be done off-line, e.g. in times when the application system is supposed to be less busy. However, if forward inferences are delayed, it may occur that queries refer to implicitly available user model contents that have not yet been inferred. So, off-line forward inferences need to be complemented with backward reasoning, so that important assumptions about the user are not lost due to a delay of forward acquisition.

4.3 Inferring Stereotyped Assumptions

In human-human interaction, people tend to form models of other people. A simple method to make a first assessment of others is to classify them and then make predictions about them according to a corresponding stereotype, i.e. the standard assumptions that one makes about members of that class [McCauley *et al.*, 1980]. The use of stereotypes in computer systems that maintain models of their users was introduced by Rich with the system Grundy [Rich, 1979; Rich, 1983]. The main components of a stereotype, as Rich defines it, are [Rich, 1989]

- a body, which contains information that is typically true of users to whom the stereotype applies, and
- a set of triggers, which decides whether the stereotype applies to a user. Hence, the triggers typically refer to assumptions about the user. Often, triggers are a subset of the stereotype contents.

In the following, stereotype mechanisms that adhere to this definition will be said to employ *container stereotypes*. .

The format of the information that is contained in the body must be compatible with the format of purely individual user model contents. Then, the standard use of stereotypes is to regularly evaluate the triggers and to let the individual user model inherit the contents of a stereotype if its triggers are satisfied.

The act of applying a stereotype to an individual user (also called stereotype *activation* [Kobsa, 1990]) is the result of an inference process. [Chin, 1993] characterizes stereotypes as a grouping of inference rules into two rules of the form

$$\text{triggers} \rightarrow \text{stereotype applies}$$
$$\text{stereotype applies} \rightarrow \text{body}$$

This is in accordance with Rich, who defines a stereotype-based inference as being the result of a two-step reasoning process:

1. If the trigger is satisfied, infer that the user is a member of the group defined by the stereotype.
2. On the basis of the stereotype, infer that the contents of the stereotype are valid assumptions about the user.

There are few user modeling systems that actually use sets of rules or other expressions of their standard formalism to represent stereotyped assumptions. The reasoning capabilities of the formalism are then used for stereotype inferences. Examples are ANATOM-TUTOR [Beaumont, 1994], which makes use of production rules (cf. Section 3.2.3), and the modal description logic $\mathcal{FALCM}$ [Hustadt, 1995], which allows for the representation of personalized stereotypes of several agents (cf. Section 3.4.2).

Stereotype reasoning is widely employed for user model acquisition. The container method, as described above, is used for example in HAM-ANS, where the user profile is constructed out of several stereotypes [Morik, 1989], and by [Moore and Paris, 1992]. An extension to the container approach as described above is to represent not only sufficient conditions for the application of a stereotype in form of triggers, but also necessary conditions, as in [Kobsa, 1990] and [Shifroni and Shanon, 1992]. The evaluation of necessary conditions may lead to the *deactivation* or *retraction* of stereotypes. This means that the stereotype contents are no longer inherited by the user model. Note that stereotype retraction is a nonmonotonic inference.

Furthermore, stereotypes can be organized in inheritance hierarchies, so that a subsumption relation between user classes is introduced. In Grundy, stereotype hierarchies are used. Another example is [Ardissono and Sestero, 1996], where a stereotype is a set of formulas of the rich modal logic that is employed for user model representation.

In many user modeling systems, assumptions about the user are relative to system domain knowledge. Therefore, a further development of stereotype reasoning was led by the idea that not only possible assumptions about the user can be grouped according to a classification of potential users, but also system knowledge can be grouped according to certain criteria. KNOME, the user modeling component of the "Unix Consultant" UC, introduced this *double-stereotype* approach [Chin, 1986; Chin, 1989]. In KNOME, users are categorized at different levels of expertise—novice, beginner, intermediate, and expert—and system knowledge (i.e., UNIX commands) is grouped according to its difficulty into four levels: simple, mundane, complex, and esoteric. Then, the basis for stereotype inferences is a relation between user stereotypes and difficulty levels. This relation vaguely states, how many of the commands of one difficulty level are known to a member of one user category, using one of the values ALL, MOST, AFEW, and NONE. So, after a user has been classified into one of the categories, the likelihood of her knowing a certain UNIX command can be inferred. The classification of a user is also based on likelihoods (KNOME actually uses a fuzzy logic approach). Based on a set of inference rules, KNOME draws primary inferences from observed user-system interactions (cf. Section 4.1.1). If an assumption that the user does (not) know a command is derived, KNOME incrementally determines likelihood values of the user's belonging to a category with the help of the difficulty of the command and the stereotype-difficulty relation.

Jameson further worked out the double-stereotype idea, based on insights from psychology [Jameson, 1992]. Also in his work, the difficulty of knowledge items plays an important role. However, no explicit categorization of users and knowledge items is employed. Relations between user groups, item characteristics and statements about the individual user are all represented in a distributed fashion within a Bayesian network.

In the generic tutoring system SMMS [Huang *et al.*, 1991], the concepts of the teaching domain are thematically grouped into *default packages*. A default package is a set of related propositions covering a subarea of the teaching domain. For each package, the user can be classified to be novice, average, or expert. So, the user can be categorized to be familiar with one subdomain and to have not yet learned the concepts of another subarea. For each of the stereotypes, a default package defines the subset of its propositions that is known to the stereotype (called the set of *defaults* of the stereotype), and the stereotype triggers with respect to this package. Default packages are arranged in a network to express relationships between subdomains. These hierarchical relationships are exploited as follows: the defaults of a stereotype within one default package help to determine the stereotype activations of its subpackages, and stereotype triggers in a package can refer to the activations of subpackages.

Stereotype systems like KNOME and SMMS that employ a grouping of domain knowledge typically have a simple set of stereotypes that reflect the linear advancements which can typically be made in a help or learning environment. The difficulty grouping of KNOME mainly helps to compact the stereotype inferences that can be made. In contrast, the subdomain grouping of SMMS reflect the fact that learners' progresses are not homogeneous with respect to the different aspects of the taught material. However, in the general scenario of interactive computer systems, the *linear stereotyping* of KNOME and SMMS, which has been utilized in other systems, too, will not always be applicable. For example, in an information system, there may be different types of users with different, unrelated informational needs and interests. Moreover, both difficulty and thematical grouping of system knowledge establish a relationship between user knowledge and system knowledge about the domain. Other assumption types like goals or preferences are difficult to integrate in such a stereotyping approach. Container stereotypes appear to be a perhaps less elegant, but more flexible means in such cases.

4.4 Logic-Based Reasoning Services

In the previous sections of this chapter, the main reasoning tasks of a user modeling system were demonstrated: Forward reasoning supports active acquisition of implicit user model contents, and backward reasoning permits access to implicit assumptions on demand of applications or the user modeling system itself. For-

primary	forward:	KNOME [Chin, 1989]
		GUMAC [Kass, 1991]
	backward:	/
mixed	forward:	[Zukerman and McConachy, 1993]
	backward:	[Zukerman and McConachy, 1993]
secondary	forward:	SMMS [Huang *et al.*, 1991]
		GUMAC [Kass, 1991]
		UMFE [Sleeman, 1985]
		[Ardissono and Sestero, 1996]
	backward:	[Appelt and Pollack, 1992]

Table 4.1: Meta-level inferences in user modeling systems

secondary	forward:	GUMAC [Kass, 1991]
	backward:	[Quilici, 1989; Quilici, 1994]

Table 4.2: Simulative inferences in user modeling systems

ward inferences can be primary, i.e., they derive assumptions from observations and other information that is external to the user modeling knowledge base, but also secondary when inferences involve internal information only. Secondary inferences simulate the user's inferences if they are based on rules or other items that are assumed to be part of the user's beliefs, goals, etc. In most cases, however, they are meta-level inferences based on acquisition rules and other meta-knowledge that is present in the UMKB. Forward inferences are typically executed when new observations or new user model contents become available, but in principle they can be delayed for off-line execution. Backward inferences are, in most cases, driven by queries that an application system poses to a user modeling system, but can also be exploited by the user modeling system itself [Quilici, 1989; Zukerman and McConachy, 1993]. Typically, backward inferences are secondary.

Tables 4.1 and 4.2 classify the systems that have been mentioned in this chapter according to the kind of inferences they employ. While Table 4.1 lists systems with meta-level inferences, Table 4.2 mentions systems with simulative inferences. In both tables, further classification dimensions are primary vs secondary (vs mixed primary and secondary) inferences, and backward vs forward inferences. Table 4.2 is very short: None of the systems described before does simulative primary inferences. This is not surprising. It seems at least unintuitive to ascribe to the user rules which she would herself use to infer assumptions about her based on her activities.

The inference mechanisms of formalisms that are employed for representation of UMKB contents will deal with secondary inferences by definition. They should provide access to implicit user model contents by either forward or backward reasoning. If a sound backward reasoning mechanism is present, forward reasoning is theoretically not needed for secondary acquisition. However, secondary forward inferences may be beneficial concerning efficiency.

Inference procedures belong to the more sophisticated facilities of a representation system. In addition, there are more basic functionalities that a system for user model representation should provide, like making an entry into the user model and make simple "database" queries, i.e. queries without backward inferences. So, what are the necessary services a (logic-based) user modeling shell system should offer, as far as access to user modeling knowledge and reasoning about user modeling knowledge is concerned? In [Russell and Norvig, 1995], the following *knowledge base access functions* are minimally required of logic-based reasoning systems:

tell adds a new fact (sentence, formula, etc.) to the knowledge base. **tell** may additionally derive some of the facts implied by the conjunction of the knowledge base and the new fact; in brief: it may do forward reasoning.

ask decides if a queried expression is entailed by the knowledge base, perhaps using backward reasoning. So, **ask** returns something like "yes" or "no" at least, but may also return justifications of the query or possible substitutions for query variables. A restricted version of **ask** will not employ reasoning mechanisms and simply decide if a queried expression is explicitly stored in the knowledge base.

tell and **ask** constitute a minimal set of service functions. They can involve the following subfunctionalities:

store adds a new fact to the knowledge base.

forward derives some of the facts implied by the conjunction of the knowledge base and an additional (perhaps new) fact.

fetch decides if a query is explicitly stored in the knowledge base [3].

derivable decides whether a queried expression can be derived from the knowledge base.

In addition, a **consistent** function may constitute another subfunctionality of **tell** that checks whether a new fact is consistent with the knowledge base.

[3] In the naming of store and fetch, we follow [Russell and Norvig, 1995].

The above list of **tell** and **ask** subfunctions almost match the requirements of user modeling systems: they allow for forward and backward inferences and additionally offer the possibility of database access to user models. Even stereotype inferences may be supported by these functions, if stereotypes and stereotype triggers are integrated into the user model representation formalism, like in Hustadt's $\mathcal{FALCM}$. In case of container stereotypes, special mechanisms are needed, which have to be integrated into the overall representation and reasoning system.

In a user modeling system, it should be clearly defined whether or not **tell** and **ask** involve forward and backward reasoning. A user modeling shell that is aimed at providing powerful representation and full access to reasoning mechanisms should implement **tell** as 'store and forward' and **ask** as 'if not fetch then derivable'. However, a shell that is aimed at offering flexibility to the developer of an application system would ideally let the developer decide about when reasoning mechanisms used. In the next chapter, the AsTRa framework for powerful and flexible logic-based user model representation and reasoning will be specified. It will be illustrated how tell, ask and other functions can be flexibly realized in a system that is based on the dichotomy of assumption types and assumption contents. Furthermore, it will be shown that container stereotypes can be integrated in such a system quite easily.

Part II

A Logic-Based Representation Framework for User Modeling Shells: Theory and Implementation

Chapter 5

AsTRa: A Representation Framework for User Modeling Shells

5.1 Assumption Types and Contents in User Modeling Knowledge Bases

In the previous chapters it was pointed out that in logic-based user modeling, the main part of a user modeling component is a deductive knowledge base. I.e., all knowledge that is involved in the user modeling process is represented with logical formalisms and stored in a user modeling knowledge base (UMKB). In addition, this knowledge is often acquired and extended via deductive inference mechanisms.

Furthermore, user model contents have been defined as consisting of two components, namely assumption type and assumption content. Two main tendencies in handling this dichotomy have been observed: On the one hand, systems often do not represent type information in the representation formalisms used. In this case, only the assumption content is formalized, while type information is expressed by storing the assumption content in a type-specific knowledge base. In the simplest case, this approach is pursued if assumptions of only one type are to be represented. But there are also systems that employ several type-specific knowledge bases within one global user modeling knowledge base [Lehman and Carbonell, 1989; McCoy, 1989; Sarner and Carberry, 1992; Kobsa *et al.*, 1994]. On the other hand, logical formalisms can be used to express both type and content information of a user model content in an integrated way. Several user modeling systems have employed predicate calculus with special predicates for assumption type information [Moore and Paris, 1992; Appelt and Pollack, 1992; Quilici, 1994] or modal and epistemic logics with their operators [Hustadt, 1995; Ardissono and Sestero, 1996] in order to represent both aspects of user model contents in one formalism.

In accordance with these tendencies, two principal approaches for representing assumption type information have been pursued in the user modeling literature and, more generally, in the literature on belief modeling. First, there is the partition approach, which distinguishes assumptions of different types by storing them into separate knowledge bases; second, there is the modal logic approach, which uses the operators of modal or epistemic logic to deal with assumption type information.

Both approaches were discussed in Section 2.4 as candidate knowledge representation paradigms for user modeling shell systems. We stated that a user modeling shell should ideally provide representation and reasoning facilities that are powerful but also flexible in order to be useful for a wide range of user modeling systems. The partition approach was found to have greater potentials as far as flexibility is concerned, while modal logics offer more reasoning power and are based on a clear semantics.

In this chapter, a framework for logic-based user model representation is presented that mainly follows the partition approach but integrates modal logic representation and reasoning. It is based on the separation of assumption type and assumption content in user model contents; basically, it permits representation of assumption type expressions. Hence, the framework is called *Assumption Type Representation* framework (AsTRa framework in brief). Within an AsTRa user modeling knowledge base, assumption types are identified with partial knowledge bases that are similar to partitions. Within assumption types, several knowledge representation formalisms may be used to represent assumption contents. Such *content formalisms* are required to be logic-based. First, this means that they provide standard knowledge base services and reasoning mechanisms that are employed to establish overall services and mechanisms of an AsTRa UMKB. Second, logic-basedness of content formalisms, together with a strict definition of the range of possible assumption types, permits a semantic characterization of AsTRa UMKB contents in terms of modal logic. In the *basic* framework as sketched so far, every single reasoning process is confined to one of the available assumption types. However, this deficiency can be overcome. Based on the relationship between basic AsTRa and modal logic, an *extended* framework can be specified that offers full modal logic representation and reasoning capabilities and integrates these with the basic facilities, thus providing means for reasoning beyond assumption types.

However, the AsTRa framework was developed not only to overcome deficiencies of the partition approach. Its main aim is to serve as a theoretical foundation of user model representation and reasoning. It is particularly designed for user modeling shell systems by providing both powerful facilities and, which is most important, flexibility with respect to the use of representation and reasoning mechanisms of a user modeling system. As to flexibility, the most important feature of a basic AsTRa system is that it may provide several formalisms for assumption content representation which the developer of a user modeling system can choose from. In an extended AsTRa system, modal logic reasoning is available, but is supposed to be used as possible alternative to the basic mechanisms only. So, the flexibility of using several content formalisms within assumption type knowledge bases is retained. However, in an extended system, flexibility will not only concern basic representation facilities. One important class of assumptions that cannot be handled within the basic AsTRa framework are *negative assumptions*, i.e., assumptions about what the beliefs or goals etc. of a user are not. *Negative assumption types* will be introduced as a specialized means for maintaining such assumptions. They can be used instead of full modal reasoning, if their expressiveness is sufficient, but can also be utilized together with modal logic expressions.

5.2 Basic Assumption Type Representation

5.2.1 Assumption Types are Partial Knowledge Bases

An assumption type is similar to a partition in being a partial knowledge base that contains all user modeling knowledge of one kind, e.g., all assumptions about user preferences, all system domain knowledge, or all mutually believed user goals. In the partition approach and also in other work on belief modeling like [Taylor *et al.*, 1996], a quite standard way of labeling assumption types is used, which we will adopt and extend for general user modeling purposes (cf. Section 2.1). Thus, an AsTRa system will be able to represent a wide range of assumption types, as it is desirable for a user modeling shell.

In AsTRa, an assumption type label is a sequence of agent-modality pairs. Possible agents are S (the system) and U (the user), and there is at least one modality B (for belief). Typically, the set of modalities is extended with W (for wants/goals), but other modalities are also possible. An AsTRa implementation will probably come up with a standard set of modalities and allow developers of user modeling systems to add further modalities, e.g. I for "interests", to this set. In principle, all modalities can be used with agents S and U. However, with the exception of B, most of them (like interests, preferences, etc.) will typically be related to the user only. Together with B, the special agent M can be used to express mutual belief (cf. Section 2.1). All assumption type labels start with the pair SB. This is to stress that all user model contents are system assumptions which are not claimed to objectively hold in the world.

Assumption type labels can be read intuitively, e.g., SBMBUW stands for "the System Believes that it is Mutually Believed that the User Wants". Other examples of possible assumption types are [Kobsa and Pohl, 1995]:

SBUB: the "privately" held assumptions of the system about the user's beliefs about the application domain, including those user beliefs which the system does not share (i.e., the user's misconceptions),

SBMBUB: the mutual beliefs of the system and the user about the user's beliefs about the application domain, including those user beliefs which the system does not share (i.e., the mutually known user misconceptions),

SBMB: the mutual beliefs of the system and the user about the application domain,

SBUW: the "privately" held assumptions of the system about the user's goals with respect to the application domain

The assumption type SB is somewhat special. It is supposed to contain system knowledge that is needed within a user modeling component. I.e., it may contain beliefs of the system about the user that do not concern her beliefs, goals, preferences, etc. Examples are the user's past actions or information about her personal situation. Also general domain knowledge that is unrelated to the user but is relevant to the user modeling process may be represented within SB. For instance, SB may contain conceptual domain knowledge, upon which inferences about the user's expertise in this domain are based.

More formally, the notion of an assumption type is defined as follows.

Definition 5.1 (assumption type) *An* assumption type *is a pair* $\langle AT, KB_{AT} \rangle$ *where*

1. $AT = a_1 m_1 \dots a_n m_n$ *is the assumption type label. AT is a sequence of pairs of agents a_i and modalities m_i, which must satisfy the following conditions:*
   - $n \geq 1$
   - $a_i \in \{S, U, M\}$
   - $m_i \in \mathcal{MOD}$ *with* $\{B\} \subseteq \mathcal{MOD}$
   - $a_1 = S$, $m_1 = B$
   - $a_i = M$ *only if* $m_i = B$

   The set of modalities $\mathcal{MOD}$ may vary between different instantiations of the AsTRa framework, which will be called "AsTRa systems". See Definition 5.10 for further specifications.

2. KB_{AT} *is the assumption type knowledge base. On an abstract level, it can be seen as a set of expressions of knowledge representation formalisms.*

Further-on, an assumption type will usually be referred to by its assumption type label AT.

5.2.2 Stereotypes

The AsTRa framework also allows for the representation of stereotypes [Rich, 1979], i.e. sets of predefined assumptions about members of a potential user group (cf. Section 4.3). However, stereotypes do not constitute distinct assumption types. If a user is identified as a member of a given user group, the stereotyped assumptions about this group become part of the system assumptions about the user. So, stereotyped assumptions belong to assumption types like SBUB, SBUW, etc., depending on whether they make a statement about potential beliefs, goals or other attitudes of group members. An AsTRa implementation needs to provide means for dynamically adding and removing stereotype contents to and from the knowledge bases of the appropriate assumption types, which can be used to implement stereotype activation and retraction.

The following definition gives a formal account of how stereotypes are integrated into the AsTRa framework. It is in line with the container approach to stereotype representation. In AsTRa, however, several containers may exist for each stereotype, each of which is associated with a different assumption type. Details of procedures for stereotype activation and retraction are not specified here, since trigger evaluation in the container approach is not a genuine part of reasoning with UMKB formalisms and hence not part of the AsTRa framework.

Definition 5.2 (stereotype) *A* stereotype S *is a set of pairs,*

$$S = \{\langle KB_1, AT_1\rangle, \ldots, \langle KB_n, AT_n\rangle\}$$

where the KB_i are pairwise different knowledge bases and the AT_i are pairwise different assumption types. As long as a user is considered member of the user group that is associated with S, the contents of a stereotype knowledge base KB_i will be part of the assumption type knowledge base KB_{AT_i}.

5.2.3 Logic-Based Assumption Contents

Assumption types are partial knowledge bases in the user modeling knowledge base of an AsTRa system. Knowledge representation formalisms are needed to represent assumption contents within the assumption type knowledge bases. The plural in "formalisms" is used intentionally: An AsTRa system may provide several formalisms for representing assumption contents. Each content formalism is required to be *logic-based*. Within the AsTRa context, there are two approaches to defining this notion, a practical one and a theoretical one. The practical approach is to require that a content formalism provides a set of knowledge base access functions that are standard in logic-based knowledge representation. The theoretical approach is to require that syntax and semantics of a content formalism can be related to first-order logic. In order to become employed for the representation of assumption contents in an AsTRa implementation, a formalism will have to satisfy the practical requirements. This means that it must offer basic functionalities of logical reasoning systems, upon which global UMKB access mechanisms can be based. However, only if theoretically logic-based content formalisms are assumed, it is possible to characterize a basic AsTRa system in terms of modal logic. In an extended AsTRa system (see Section 5.3), modal logic reasoning will be able to consider such user model contents only that are represented in a theoretically logic-based formalism.

In general, a content formalism F consists of at least two components: First, it specifies a language $\mathcal{L}_F$, which is used to formulate sentences or expressions the formalism can deal with. Second, there is a derivability relation $\vdash_F$ that determines what can be derived from a F knowledge base.

Definition 5.3 (content formalism) *An AsTRa* content formalism *F is a pair $\langle \mathcal{L}_F, \vdash_F \rangle$, where*

- *$\mathcal{L}_F$ is the* language *of F. It specifies the range of possible expressions of F (briefly called F-expressions), and*
- *$\vdash_F$ is the* derivability relation *of F, which determines for all knowledge bases KB and expressions ϕ of F whether ϕ can be derived from KB (KB $\vdash_F \phi$).*

The AsTRa framework allows the use of several content formalisms in parallel. Then, within assumption type knowledge bases KB_{AT}, there are "sections" for each formalism F, each of which constitutes a separate knowledge base that contains expressions of F only. It is required that for any assumption content ac a unique formalism $F(ac)$ can be determined such that $ac \in \mathcal{L}_{F(ac)}$; this formalism will then be employed to handle this content expression. However, content formalisms are not restricted to process the contents of their own section of an assumption type knowledge base only. In an AsTRa implementation, the reasoning procedures of one content formalism may consider assumption contents of other formalisms, in order to provide hybrid reasoning capabilities. As a consequence, in the following specifications, the reasoning procedures of content formalisms will always be applied to KB_{AT} as a whole.

Figure 5.1 abstractly illustrates an assumption type with three content formalisms F_1, F_2, and F_3. Each formalism is responsible for a section of the assumption type knowledge base. The dashed lines indicate, however, that the border between sections is open for the formalisms to take contents of other sections into account. Note that the "associated information" that is mentioned in the figure will be explained in Chapter 6 (see also Table 7.1).

Practical Requirements

A practically usable representation formalism F offers a set of functions that allows to deal with knowledge bases that contain expressions of $\mathcal{L}_F$. According

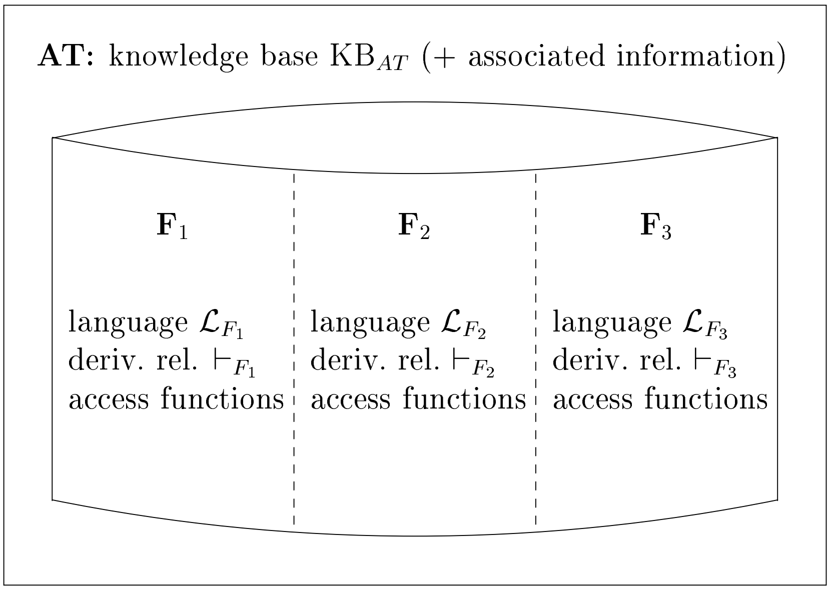


Figure 5.1: Assumption type with content formalisms

to [Russell and Norvig, 1995, p.152], for a logical knowledge base, "there must be a way to add new sentences to the knowledge base, and a way to query what is known". That is, a formalism mainly needs to provide access functions like *tell*[1] and *ask* for input and queries to knowledge bases. In Section 4.4, it was already noted that in logic-based reasoning systems such access functions may be more specific like *store* and *fetch* (for data-base-like entries and retrieval), *forward* and *derivable* (for forward- and backward-directed inferences) and *consistent* (for checking whether an expression is consistent with a knowledge base). An AsTRa content formalism F is required to provide the latter five functions for access to assumption contents *ac* within an assumption type knowledge base. This access interface to assumption type knowledge bases will be employed by an AsTRa system for global AsTRa knowledge base access.

Definition 5.4 (access functions of an AsTRa content formalism)
An AsTRa content formalism F provides a set of boolean-valued functions, namely $store_F$, $forward_F$, $consistent_F$, $fetch_F$, and $derivable_F$. Let ac be an expression of F (i.e., $ac \in \mathcal{L}_F$), and let KB_{AT} be an assumption type knowledge base, then

$store_F(KB_{AT}, ac)$ *adds ac to KB_{AT} and returns* **true***.*

$forward_F(KB_{AT}, ac)$ *computes some of the expressions ac' that can be derived*

[1]Content formalism access functions will be written in italics in order to distinguish them from global UMKB access functions, which (as already done in Section 4.4) will be written in bold font.

from the conjunction of KB_{AT} *and ac, (i.e.,* $KB_{AT} \cup \{ac\} \vdash_F ac'$*), invokes* $store_F(KB_{AT}, ac')$ *on them and returns* **true**.

$consistent_F(KB_{AT}, ac)$ *returns* **true** *iff ac is consistent with* KB_{AT} *(i.e., there is a model for* $KB_{AT} \cup \{ac\}$*).*

$fetch_F(KB_{AT}, ac)$ *returns* **true** *iff ac is explicitly stored in* KB_{AT}.

$derivable_F(KB_{AT}, ac)$ *returns* **true** *iff ac is derivable from* KB_{AT} *(i.e.,* $KB_{AT} \vdash_F ac$*).* $derivable_F$ *is an extension to fetch: If* $fetch_F(KB_{AT}, ac) = \mathbf{true}$ *then also* $derivable_F(KB_{AT}, ac) = \mathbf{true}$

A formalism with the above set of functions can be used quite flexibly. In case of entries to a knowledge base, it can be decided if a consistency check or forward reasoning is appropriate, and in case of queries, it can be chosen between retrieval of explicit contents and derivation of implicit contents.

Theoretical Requirements

In a theoretical sense, a representation formalism F can be called logic-based if it has a semantics which defines a notion of truth, thus giving a meaning to each expression of $\mathcal{L}_F$. Based on truth, an entailment relation is set up; the derivability relation $\vdash_F$ is the syntactic variant of semantic entailment. From an abstract point of view, sets of $\mathcal{L}_F$-expressions form a knowledge base KB. The entailment relation determines what "follows" from such a knowledge base, i.e. which knowledge is implicitly contained in KB in addition to its explicitly contained elements. The reasoning facilities of the formalism, i.e., the derivability relation and its implementations, must be related to this semantic notion. A minimal requirement is that inferences are sound with respect to semantic entailment; a maximal and perhaps sometimes undesirable requirement is that inferences are also complete.

Within the AsTRa framework, the notion "logic-based" is defined more strictly. Logic-based content formalisms must be closely related to first-order predicate calculus. It is required that on the syntactic level, F-expressions can be translated into FOPC. In order for this translation to make sense, also the semantics of F must correspond with those of FOPC. The consequence of these demands is that a logic-based formalism can be identified with (a subset of) first-order logic.

Definition 5.5 (logic-based content formalism) *A* logic-based content formalism F *is a triple* $\langle \mathcal{L}_F, t_F, \vdash_F \rangle$*, where:*

- $\mathcal{L}_F$ *is the* language *of* F.
- t_F *is a* translation function *that translates assumption contents* $ac \in \mathcal{L}_F$ *into FOPC formulas:* $t_F(ac) \in \mathcal{L}_{FOPC}$*. When applied to assumption type knowledge bases* KB_{AT}*, it translates all* F*-expressions of* KB_{AT}*:*

$$t_F(KB_{AT}) := \{t_F(ac) \mid ac \in (KB_{AT} \cap \mathcal{L}_F)\}$$

- *The* derivability relation $\vdash_F$ *must be sound with respect to FOPC:*

$$KB_{AT} \vdash_F ac \implies t_F(KB_{AT}) \models t_F(ac)$$

where ac is an F-expression, and $\models$ is the entailment relation of FOPC.

- *The $derivable_F$ function of the formalism implements the derivability relation $\vdash_F$: $derivable_F(KB_{AT}, ac) = \textbf{\textit{true}} \iff KB_{AT} \vdash_F ac$.*

As far as syntax is concerned, assumption contents and hence assumption type knowledge bases can be identified with FOPC expressions and knowledge bases respectively:

Definition 5.6 (standard translation) *Let $F(ac)$ denote the content formalism of an assumption content ac, i.e., $ac \in \mathcal{L}_{F(ac)}$. The* standard translation *of assumption contents and assumption type knowledge bases, τ, is defined as follows:*

1. $\tau(ac) := t_{F(ac)}(ac)$
2. $\tau(KB_{AT}) := \{\tau(ac) \mid ac \in KB_{AT}\}$

Note that the soundness requirement of Definition 5.5 extends from partially translated knowledge bases $t_F(\mathrm{KB})$ to fully translated ones, $\tau(\mathrm{KB})$:

$$\mathrm{KB}_{AT} \vdash_F ac \implies \tau(\mathrm{KB}_{AT}) \models \tau(ac) \tag{5.1}$$

The reason is that every FOPC model of $\tau(\mathrm{KB}_{AT})$ is also a model of $t_F(\mathrm{KB}_{AT}) \subseteq \tau(\mathrm{KB}_{AT})$. According to Definition 5.5, also $t_{F(ac)}(ac) = \tau(ac)$ is satisfied by these models and hence entailed by $\tau(\mathrm{KB}_{AT})$.

Due to these correspondences, assumption contents ac and assumption type knowledge bases KB_{AT} can be identified with their standard translations, $\tau(ac)$ and $\tau(\mathrm{KB}_{AT})$ respectively. In the following, we will let ac and KB_{AT} denote both non-translated and translated assumption contents and assumption type knowledge bases respectively, unless the distinction is relevant.

For the characterization of AsTRa in terms of modal logic, an even stronger correspondence between content formalisms and predicate calculus is needed. A content formalism F will be most strongly related to FOPC, if its $derivable_F$ function is sound *and complete* with respect to FOPC entailment, taking the whole assumption type knowledge base into account:

Definition 5.7 (ideally logic-based content formalism) *A logic-based content formalism F is called* ideally logic-based*, if and only if*

$$derivable_F(KB_{AT}, ac) = \textbf{\textit{true}} \iff KB_{AT} \models ac.$$

This is a very strong notion, since the $derivable_F$ function of an ideally logic-based formalism F is required to consider all contents of KB_{AT}, including expressions of other formalisms. A weaker, but nevertheless interesting property can be ascribed to a formalism F if its derivation function processes *all F expressions* in an assumption type knowledge base equivalently to FOPC. Let $\mathrm{KB}_{AT/F}$ denote the set of all F expressions in KB_{AT} (again, we identify this set with its FOPC translation $\tau(\mathrm{KB}_{AT/F}) = t_F(\mathrm{KB}_{AT/F})$):

Definition 5.8 (F-ideally logic-based content formalism) *A logic-based content formalism F is called* F-ideally logic-based*, if and only if*

$$derivable_F(KB_{AT}, ac) = \textbf{\textit{true}} \iff KB_{AT/F} \models ac.$$

5.2.4 Using an AsTRa UMKB

So far, assumption types, stereotypes, and assumption content formalisms have been discussed. In this section, we will show how these ingredients are composed into an AsTRa system with a coherent user modeling knowledge base (UMKB), and how this UMKB can be used. An AsTRa knowledge base consists of assumption types (i.e., of assumption type knowledge bases), plus a number of stereotypes. Both notions were defined in previous sections. An AsTRa system can then be defined to consist of at least one UMKB and a set of content formalisms that can be utilized for representation and reasoning within assumption types. For using an AsTRa UMKB, global **tell** and **ask** functions are provided. These functions can be used on different reasoning levels which determine the complexity of the reasoning processes employed. In a basic AsTRa system, the global functions will always refer to type-internal mechanisms. I.e., they will invoke access functions of content formalisms and apply them to the appropriate assumption type knowledge bases.

Definition 5.9 (basic AsTRa knowledge base) *An* AsTRa knowledge base *is a pair $\langle \mathcal{AT}, \mathcal{S} \rangle$, where*

- *$\mathcal{AT} = \{AT_1, \ldots, AT_n\}$ is a set of assumption types, and*
- *$\mathcal{S} = \{S_1, \ldots, S_m\}$ is a set of stereotypes.*

Definition 5.10 (AsTRa system) *An* AsTRa system *is a quadruple $\langle \mathcal{UMKB}, UMKB_c, \mathcal{F}, \mathcal{MOD} \rangle$, where*

- *$\mathcal{UMKB}$ is a set of AsTRa knowledge bases. That is, an AsTRa implementation may maintain several UMKBs, e.g., for several users in parallel.*

- *$UMKB_c$ is a pointer that refers to one of the AsTRa knowledge bases in $\mathcal{UMKB}$, which is called the "current" knowledge base (hence the subscript "c"). At any time, only the current knowledge base can be accessed. AsTRa implementations with several UMKBs need to implement a mechanism for switching between the knowledge bases.*

- *$\mathcal{F} = \{F_1, \ldots, F_r\}$ is a set of logic-based content formalisms.*

- *$\mathcal{MOD}$ is a set of modalities that are allowed in assumption type labels. $\mathcal{MOD} \supseteq \{B\}$, i.e., there is a belief modality at least[2].*

Together, assumption types and assumption contents, which may partly originate from stereotypes, constitute a user modeling knowledge base in an AsTRa system. Also for such a global knowledge base, procedures are needed to access and manipulate knowledge base contents. In a basic AsTRa system, a UMKB content is an assumption content that is stored in an assumption type knowledge base. Hence, they are denoted $AT{:}ac$, where AT is an assumption type label, and ac is an assumption content expression. ac is an expression of a content formalism $F(ac) \in \mathcal{F}$, i.e., $ac \in \mathcal{L}_{F(ac)}$. Global knowledge base access functions that deal with such *type-internal* $AT{:}ac$ expressions will recur to the "local" functions of the appropriate content formalism $F(ac)$ and apply them to the appropriate assumption type knowledge base KB_{AT}.

The AsTRa framework defines two central knowledge base access functions, namely **tell** and **ask**, which will be composed of more fine-grained functions. Among these subfunctions are **store** and **fetch**, which serve as "database functions" for simply entering and retrieving AsTRa expressions to and from a UMKB.

Definition 5.11 (AsTRa store and fetch) *The AsTRa access functions* **store** *and* **fetch** *are defined as follows:*

- **store**($UMKB, AT{:}ac$) $:= store_{F(ac)}(KB_{AT}, ac)$

- **fetch**($UMKB, AT{:}ac$) $:= fetch_{F(ac)}(KB_{AT}, ac)$

One of the central goals of the AsTRa framework is to provide flexible representation and, in particular, reasoning mechanisms. The user of an AsTRa system shall always be able to choose from the available facilities. Representation mechanisms are determined by choosing from content formalisms when formulating type-internal expressions. As to reasoning facilities, we will distinguish between different *reasoning levels*, which employ different reasoning mechanisms. That is, at each level the AsTRa reasoning functions **derivable**, **forward**, and **consistent** (cf. Section 4.4) will be defined differently. An AsTRa implementation is

[2]Since assumption type labels are required to start with SB (cf. Definition 5.1), the modality B must be present in any AsTRa system.

required to provide facilities for access to the UMKB that permit to choose from the available range of reasoning levels.

For each level L, reasoning functions $\mathbf{derivable}_L$, $\mathbf{forward}_L$, and $\mathbf{consistent}_L$ must be defined.

Definition 5.12 (reasoning level) *A* reasoning level *L is determined by a triplet of AsTRa UMKB reasoning functions*

$$\langle \mathbf{derivable}_L,\ \mathbf{forward}_L,\ \mathbf{consistent}_L \rangle$$

A reasoning level is called type-internal*, when it deals with type-internal expressions $AT{:}ac$ only.*

Now, global **tell** and **ask** functions can be defined. **tell** will enter an assumption into a UMKB and perform forward inferences, if the assumption is consistent with the UMKB. **ask** verifies if an assumption is contained in a UMKB, using backward inferences if the assumption is not an explicit content. Both **tell** and **ask** take an additional parameter L, which determines the reasoning level to be used:

Definition 5.13 (AsTRa tell and ask) *The global knowledge base access functions of an AsTRa system,* **tell** *and* **ask***, are defined as follows:*

1. $\mathbf{tell}(L, UMKB, AT{:}ac) :=$
   if $\mathbf{consistent}_L(UMKB, AT{:}ac)$ *then*
   $\mathbf{store}(UMKB, AT{:}ac)$ *and* $\mathbf{forward}_L(UMKB, AT{:}ac)$
2. $\mathbf{ask}(L, UMKB, AT{:}ac) :=$
   $\mathbf{fetch}(UMKB, AT{:}ac)$ *or* $\mathbf{derivable}_L(UMKB, AT{:}ac)$

5.2.5 Reasoning Levels in a Basic AsTRa System

In a basic AsTRa system with its assumption type knowledge bases and content formalisms, the reasoning functions of content formalisms can be employed to define the behavior of global reasoning functions. First, however, we can define a minimal reasoning level that does no reasoning at all. I.e., reasoning functions will be reduced to database functions. The minimal level is labeled 0.

Definition 5.14 (minimal reasoning level 0) *At the minimal type-internal AsTRa reasoning level 0, reasoning functions are defined as follows:*

- $\mathbf{derivable}_0(UMKB, AT{:}ac) := \boldsymbol{false}$
- $\mathbf{forward}_0(UMKB, AT{:}ac) := \boldsymbol{true}$
- $\mathbf{consistent}_0(UMKB, AT{:}ac) := \mathbf{true}$

Note that **tell** and **ask** are defined as logical combinations of reasoning functions. Hence, at reasoning level 0, **tell** collapses to **store**, and **ask** collapses to **fetch**.

Given the possibilities defined so far, there is only one alternative to the minimal reasoning level with its database-like UMKB access, namely employment of the (type-internal) reasoning functions of content formalisms. Thus, a second reasoning level can be formed, which is the basic AsTRa reasoning level; it is confined to reasoning within one assumption type. This second level will be labeled TI (short for "type-internal"). In the extended AsTRa framework, new reasoning levels will be introduced that also deal with type-internal expressions. However, at those levels, contents of more than one assumption type will be involved in reasoning processes, so that the label TI is truly appropriate for the basic reasoning level that is defined now.

Definition 5.15 (basic reasoning level TI) *At the basic type-internal AsTRa reasoning level, reasoning functions are defined as follows:*

- $\mathbf{derivable}_{TI}(\mathit{UMKB}, \mathit{AT}{:}\mathit{ac}) := \mathit{derivable}_{F(ac)}(\mathit{KB}_{AT}, \mathit{ac})$
- $\mathbf{forward}_{TI}(\mathit{UMKB}, \mathit{AT}{:}\mathit{ac}) := \mathit{forward}_{F(ac)}(\mathit{KB}_{AT}, \mathit{ac})$
- $\mathbf{consistent}_{TI}(\mathit{UMKB}, \mathit{AT}{:}\mathit{ac}) := \mathit{consistent}_{F(ac)}(\mathit{KB}_{AT}, \mathit{ac})$

Reasoning Levels and Simple Content Formalisms

As specified in Definition 5.4, content formalisms need to provide five knowledge base access functions. However, there will probably be representation formalisms that could well be used as content formalisms in an AsTRa system but do not provide such a fine-grained interface. Such formalisms shall not be excluded from becoming AsTRa content formalisms. For this purpose, we introduce the notion of a *simple content formalism*. A simple formalism F provides two basic access functions tell_F and ask_F only. An AsTRa system may employ a simple formalism F to represent assumption contents; then, the full AsTRa content formalism interface will be constructed for F as follows:

- $\mathit{store}_F(\mathrm{KB}_{AT}, ac) := \mathit{tell}_F(\mathrm{KB}_{AT}, ac)$
- $\mathit{forward}_F(\mathrm{KB}_{AT}, ac) := \mathit{tell}_F(\mathrm{KB}_{AT}, ac)$
- $\mathit{consistent}_F(\mathrm{KB}_{AT}, ac) := \mathbf{true}$
- $\mathit{fetch}_F(\mathrm{KB}_{AT}, ac) := \mathit{ask}_F(\mathrm{KB}_{AT}, ac)$
- $\mathit{derivable}_F(\mathrm{KB}_{AT}, ac) := \mathit{ask}_F(\mathrm{KB}_{AT}, ac)$

Note that for type-internal expressions $AT{:}ac$, where the content formalism of ac is a simple formalism, the reasoning levels 0 and TI become identical. At both levels, the simple access functions $tell_F$ and ask_F are used. The user of an AsTRa system with simple content formalisms should be informed that he can no longer control the use of type-internal reasoning mechanisms when using these formalisms.

5.2.6 Expressive and Inferential Power

In order to characterize the AsTRa framework in a formal way, this section relates the framework to representation and reasoning with modal logic. This is quite worthwhile, since first, modal logic is typically used for reasoning about beliefs and therefore has been employed for user modeling quite often (e.g., cf. [Allgayer *et al.*, 1992; Hustadt, 1995; Ardissono and Sestero, 1996]), and second, it provides a well-established formal apparatus.

A syntactical relationship between AsTRa and modal logic is quite obvious: An assumption type label

$$AT = a_1 m_1 \ldots a_n m_n$$

is a sequence of agent-modality pairs, which corresponds to a sequence of modal operators with modality and agent indices

$$\mathcal{M}(AT) := \Box_{(m_1,a_1)} \ldots \Box_{(m_n,a_n)}$$

Furthermore, any assumption content ac has a corresponding FOPC expression $\tau(ac)$. Hence, a type-internal expression $AT{:}ac$ corresponds to a modal logic formula $\mathcal{M}(AT)\,\tau(ac)$. E.g., the AsTRa expression

SBUW:printed(userdoc)

(i.e., the FOPC assumption content printed(userdoc) of type SBUW) corresponds to the modal formula

$$\Box_{(B,S)}\Box_{(W,U)}\text{printed(userdoc)}$$

The modal logic, which exactly comprises $\mathcal{M}(AT)\,\tau(ac)$ formulas for all content formalisms F (i.e., which allows $\mathcal{M}(AT)\,p$ formulas where p is a FOPC expression), is called *assumption logic* (AL). Assumption type knowledge bases KB_{AT} can be represented by sets of AL formulas, KB^{AL}_{AT}:

$$\text{KB}^{AL}_{AT} := \{\mathcal{M}(AT)\,\tau(ac) \mid ac \in \text{KB}_{AT}\}$$

If for every assumption type in an AsTRa UMKB this set is generated, the union of all these sets is a reformulation of the whole UMKB as AL formulas, which will be called UMKB^{AL}:

$$\text{UMKB}^{AL} := \bigcup_{AT \in \mathcal{AT}} \text{KB}^{AL}_{AT}$$

AL is a normal, multi-modal, multi-agent logic with a restricted formula syntax. Assuming a standard possible worlds semantics, axiom K and hence also modal modus ponens hold for for any possible operator $\Box_{(m,a)}$ of AL (cf. Section 3.4.1):

$$\Box_{(m,a)}(\Phi \to \Psi) \to (\Box_{(m,a)}\Phi \to \Box_{(m,a)}\Psi) \quad \text{(K)}$$
$$\Box_{(m,a)}\Phi \wedge \Box_{(m,a)}(\Phi \to \Psi) \to \Box_{(m,a)}\Psi \quad \text{(modal modus ponens)}$$

Furthermore, modus ponens and the necessitation rule (cf. Section 3.4.1) are available as general inference rules. Since modus ponens ($\Phi, \Phi \to \Psi \vdash \Psi$) does not involve modal operators, it remains unchanged. In case of the necessitation rule, however, we need a variant that covers AL operators:

$$\Phi \vdash \Box_{(m,a)}\Phi \quad \text{(necessitation rule)}$$

Besides the syntactical relationship, there is also a semantical relationship between AsTRa reasoning and AL, concerning their deductive power. If content formalisms are ideally logic-based, the basic AsTRa reasoning function $\mathbf{derivable}_{TI}$ constitutes a sound and complete calculus for AL. With $\models$ being the standard entailment relation, the following theorem holds:

Theorem 1 *If all content formalisms of an AsTRa system are ideally logic-based, then basic type-internal AsTRa reasoning is sound and complete with respect to assumption logic*[3]*:*

$$\mathbf{derivable}_{TI}(UMKB, AT{:}ac) = \boldsymbol{true} \iff UMKB^{AL} \models \mathcal{M}(AT)\,ac$$

Proof First, the left-hand side of the equivalence can be transformed:

$$\begin{array}{lll} \mathbf{derivable}_{TI}(\mathrm{UMKB}, AT{:}ac) = \mathbf{true} & \iff & \text{(Definition 5.15)} \\ \mathit{derivable}_{F(ac)}(\mathrm{KB}_{AT}, ac) = \mathbf{true} & \overset{(*)}{\iff} & \text{(Definition 5.7)} \\ \mathrm{KB}_{AT} \models ac & & \end{array}$$

So, we still need to prove:

$$\mathrm{KB}_{AT} \models ac \iff \mathrm{UMKB}^{AL} \models \mathcal{M}(AT)\,ac \tag{5.2}$$

In order to prove this equivalence, we need to establish relationships between (a) FOPC semantics within assumption type knowledge bases KB_{AT} and (b) AL semantics of the whole UMKB^{AL}. First, we define a relationship between interpretations: The interpretation of a modal AL formula $\mathcal{M}(AT)\,ac$ can be reduced to an FOPC interpretation of the assumption content expression ac. The information that the assumption type AT conveys is preserved by defining a subset of the possible worlds of an AL frame that is closely related to AT. More

[3] Recall that we identify assumption contents ac with their FOPC translations $\tau(ac)$.

precisely: The interpretation of $\mathcal{M}(AT)\,ac$ is determined by the interpretation of ac in the worlds that are accessible from the initial world according to the operator sequence $\mathcal{M}(AT)$ and hence according to the assumption type AT. This set of worlds will be called *AT-accessible*.

Definition 5.16 (AT-accessible worlds) *Let AT, AT' be assumption type labels and $F = \langle D, W, w_0, \mathcal{R}\rangle$ be a modal frame.*

1. *For $w \in W$, the set of (w, AT)*-accessible worlds*, $W(w, AT)$, is defined recursively:*
   (a) *$AT = a_1 m_1$: $W(w, AT) := \{w' \in W \mid (w, w') \in R_{(m_1,a_1)}\}$*
   (b) *$AT = a_1 m_1 AT'$: $W(w, AT) := \bigcup_{(w,w') \in R_{(m_1,a_1)}} W(w', AT')$*
2. *$W(w_0, AT)$ is briefly called the set of AT*-accessible worlds.

Lemma 1 *Let I be a modal interpretation function, and let AT be an assumption type. Then $I(w, \mathcal{M}(AT)\,ac) = true$ iff $I(w', ac) = true$ for all $w' \in W(w, AT)$, i.e., for all (w, AT)-accessible worlds w'.*

Proof The lemma is proven by induction over the length of AT.

1. $AT = a_1 m_1$: $I(w, \Box_{(m_1,a_1)}\, ac) =$ true
   iff $I(w', ac) =$ true for all w' such that $(w, w') \in R_{(m_1,a_1)}$
   iff $I(w', ac) =$ true for all (w, AT)-accessible worlds
2. $AT = a_1 m_1 AT'$:
   $I(w, \Box_{(m_1,a_1)}\mathcal{M}(AT')\,ac) =$ true
   iff $I(w', \mathcal{M}(AT')\,ac) =$ true for all w' such that $(w, w') \in R_{(m_1,a_1)}$
   iff $I(w'', ac) =$ true for all (w', AT')-accessible worlds w''
   for all w' such that $(w, w') \in R_{(m_1,a_1)}$
   iff $I(w'', ac) =$ true for all $w'' \in \bigcup_{(w,w') \in R_{(m_1,a_1)}} W(w', AT')$
   iff $I(w'', ac) =$ true for all (w, AT)-accessible worlds w''

$\triangleleft$

From Lemma 1 and Definition 5.16, we immediately get the following

Corollary 1 *Let $\mathcal{I} = \langle I, F\rangle$ be a modal interpretation. $\mathcal{I}(\mathcal{M}(AT)\,ac) = true$ iff $I(w, ac) = true$ for all AT-accessible worlds w.*

Based on this result, we can go one step further: We directly relate AL interpretations to FOPC interpretations with respect to an assumption type AT via the set of AT-accessible worlds. For an FOPC interpretation function I_{FOPC} and an assumption type AT, a corresponding AL interpretation function can be constructed that in all AT-accessible worlds interprets non-modal formulas like I_{FOPC}. This construction works both ways: For an AL interpretation function I_{AL} and an assumption type AT, a corresponding FOPC interpretation function can be constructed that in all AT-accessible worlds interprets non-modal formulas like I_{AL}.

Definition 5.17 (AT-corresponding interpretation)
*Let $\mathcal{I}_{AL} = \langle I_{AL}, \langle D, W, w_0, \mathcal{R}\rangle\rangle$ be an AL interpretation. Let $\mathcal{I}_{FOPC} = \langle I_{FOPC}, D'\rangle$ be a FOPC interpretation. $\mathcal{I}_{AL}$ AT-*corresponds *to $\mathcal{I}_{FOPC}$ (and $\mathcal{I}_{FOPC}$ AT-*corresponds *to $\mathcal{I}_{AL}$) iff for all AT-accessible worlds $w \in W(w_0, AT)$*

$$I_{AL}(w, ac) = I_{FOPC}(ac)$$

for all ac, and $D' \subseteq D$.

Note that there may be more than one AT-corresponding AL interpretation for a given FOPC interpretation, since their behavior is not defined for worlds that are not AT-accessible.

So far, we have established relationships between FOPC and AL interpretations. However, sentence 5.2, which is still to be proven, is about entailment. Hence, we need to relate *models* of FOPC knowledge bases (i.e., FOPC-translations of assumption type knowledge bases) and AL knowledge bases. This can be done by putting Corollary 1 and Definition 5.17 together. On the one hand, they imply that an FOPC interpretation which AT-corresponds to a model of an AL knowledge base UMKB^{AL} is a model of the assumption type knowledge base KB$_{AT}$. On the other hand, we can construct an AL model of UMKB^{AL} from models of all assumption type knowledge bases of the original AsTRa knowledge base UMKB. An AL interpretation that AT-corresponds to an FOPC model of each KB$_{AT}$ is a model of UMKB^{AL}. Such an interpretation can certainly be constructed, if the sets of AT-accessible worlds are pairwise disjoint: Assume that in a UMKB, KB$_{AT_1}$ contains p and KB$_{AT_2}$ contains $\neg p$. A model of UMKB^{AL} satisfies both $\mathcal{M}(AT_1)\, p$ and $\mathcal{M}(AT_2)\, \neg p$, and hence (following Corollary 1) satisfies p in all AT_1-accessible worlds $W(w_0, AT_1)$ and $\neg p$ in all AT_2-accessible worlds $W(w_0, AT_2)$. This is only possible if $W(w_0, AT_1) \cap W(w_0, AT_2) = \emptyset$.

According to these ideas, we define a family of *canonical models* for an AL-transformed AsTRa user modeling knowledge base UMKB^{AL}.

Definition 5.18 (canonical model) *Let UMKB= $\langle \mathcal{AT}, \mathcal{S}\rangle$ be an AsTRa knowledge base and UMKB^{AL} be the corresponding AL knowledge base. Let $\mathcal{I} = \langle I, \langle D, W, w_0, \mathcal{R}\rangle\rangle$ be a modal interpretation. $\mathcal{I}$ is called a* canonical model *of UMKB^{AL}, if and only if:*

1. *W, w_0, and $\mathcal{R}$ are chosen such that the sets of AT-accessible worlds are pairwise disjoint for all assumption types $AT \in \mathcal{AT}$;*[4]

2. *for all $AT \in \mathcal{AT}$, $\mathcal{I}$ AT-corresponds to an FOPC model of KB_{AT};*

3. *the domain D is the union of the domains of the FOPC models used in 2.*

We will now see that a canonical model deserves its name.

Corollary 2 *A canonical model of an AL knowledge base $UMKB^{AL}$ really is a model of this knowledge base.*

Proof Let $\mathcal{I} = \langle I, F\rangle$ be a canonical model of UMKB^{AL}. It is a model of UMKB^{AL} iff $\mathcal{I}(\mathcal{M}(AT)\, ac) = \text{true}$ for all formulas $\mathcal{M}(AT)\, ac \in \text{UMKB}^{AL}$. According to Definition 5.18, $\mathcal{I}$ AT-corresponds to a FOPC model $\mathcal{I}_{FOPC} = \langle I_{FOPC}, D\rangle$ of KB_{AT}. I.e., for all AT-accessible worlds, $I(w, ac) = I_{FOPC}(ac) = \text{true}$. According to Lemma 1, hence also $\mathcal{I}(\mathcal{M}(AT)\, ac) = \text{true}$. $\triangleleft$

A canonical model that AT-corresponds to a model $\mathcal{I}_{FOPC}$ of an assumption type knowledge base KB_{AT} is called *corresponding canonical model* of $\mathcal{I}_{FOPC}$.

The proof of (5.2) is now quite straightforward. For both directions of $\iff$, the entailment relation of the conclusion side is established using the entailment relation of the premise side.

$\Longrightarrow$: $\mathcal{I}_{AL} = \langle I_{AL}, F\rangle$ is a model of UMKB^{AL}

- $\Rightarrow$ $\mathcal{I}_{AL}$ satisfies $\text{KB}^{AL}_{AT} \subseteq \text{UMKB}^{AL}$
- $\Rightarrow$ $\mathcal{I}_{AL}(\mathcal{M}(AT)\, ac') = \text{true}$ for all $ac' \in \text{KB}_{AT}$
- $\Rightarrow$ (Corollary 1) for all AT-accessible worlds w, $I_{AL}(w, ac') = \text{true}$ for all $ac' \in \text{KB}_{AT}$
- $\Rightarrow$ (Definition 5.17) For an AT-corresponding interpretation of $\mathcal{I}_{AL}$, $\mathcal{I}_{FOPC} = \langle I_{FOPC}, D\rangle$, $I_{FOPC}(ac') = \text{true}$ for all $ac' \in \text{KB}_{AT}$
- $\Rightarrow$ $\mathcal{I}_{FOPC}$ is a model of KB_{AT}
- $\Rightarrow$ (premise) $\mathcal{I}_{FOPC}$ satisfies ac
- $\Rightarrow$ (Definition 5.17) for all AT-accessible worlds w, $I_{AL}(w, ac) = \text{true}$
- $\Rightarrow$ (Corollary 1) $\mathcal{I}_{AL}$ satisfies $\mathcal{M}(AT)\, ac$

$\Longleftarrow$: $\mathcal{I}_{FOPC}$ is a model of KB_{AT}

- $\Rightarrow$ for $\mathcal{I}_{FOPC}$, there is a corresponding canonical model $\mathcal{I}_{AL} = \langle I_{AL}, F\rangle$ of UMKB^{AL}

[4]This construction is possible for non-reflexive accessibility relations. So, it is assumed that the reflexivity axiom $\Box\Phi \rightarrow \Phi$ does not hold for any AL operator. This classical "knowledge axiom" is not relevant to AL, since there are no non-modal formulas and hence no notion of absolute truth.

$\Rightarrow$ (premise) $\mathcal{I}_{AL}$ satisfies $\mathcal{M}(AT)\,ac$
$\Rightarrow$ (Corollary 1) in all AT-accessible worlds w, $I_{AL}(w, ac) = \text{true}$
$\Rightarrow$ (Definition 5.17) $\mathcal{I}_{FOPC}(ac) = \text{true}$
$\Rightarrow$ $\mathcal{I}_{FOPC}$ satisfies ac.

$\triangleleft$

We have already stated that it is a strong requirement of content formalisms to be ideally logic-based. For realistic AsTRa systems, it is more likely that content formalisms are F-ideally logic-based. In this case, the equivalence $(*)$ in the transformation that was made at the beginning of the proof (p.111), does no longer hold in both directions. Because of the soundness requirements for all logic-based content formalisms (cf. Definition 5.5 and Formula 5.1), the backward implication $\Longleftarrow$ still holds, so that soundness of AsTRa reasoning with respect to AL can still be proven. In addition, a completeness statement can still be made if, like an F-ideally logic-based formalism, modal logic reasoning also considers the F expressions within an AsTRa knowledge base only. Let UMKB_F denote the subset of an AsTRa UMKB that only contains expressions of F as assumption contents; UMKB_F^{AL} denotes its AL translation. Then, a weaker version of Theorem 1 can be proven analogously:

Theorem 2 *If all content formalisms of an AsTRa system are F-ideally logic-based, then basic type-internal AsTRa reasoning is sound and complete with respect to assumption logic in the following restricted way:*

$$\mathbf{derivable}_{TI}(UMKB, AT{:}ac) = true \iff UMKB^{AL}_{F(ac)} \models \mathcal{M}(AT)\,ac$$

With the equivalence of either Theorem 1 or Theorem 2, modus ponens is unlimitedly applicable within assumption type knowledge bases, and tautologies of FOPC hold. These properties correspond to modal modus ponens and the necessitation rule, respectively. Note that these conditions are not always desirable for user modeling. AsTRa implementations may decide to abandon them, mainly by using content formalisms which are not (F-)ideally logic-based.

5.3 Extended Assumption Type Representation

5.3.1 Assumption Types and Full Modal Reasoning

In the basic AsTRa framework, different types of assumptions can be represented, and assumption contents can be expressions of logic-based formalisms. Possible assumption types are of a positive nature, i.e. only assumptions about what the user *does* believe, want, etc. are allowed, but not assumptions about what the user *does not* believe, want etc. Furthermore, assumption types are isolated; neither relationships between specific contents of different types nor general relationships between assumption types can be expressed. Although the AsTRa framework satisfies the representational and inferential needs of many user modeling systems, we wanted to enable the application developer to express relationships between different assumption types as well as negative assumptions.

In the previous section, we described how the (type-internal) contents of an AsTRa knowledge base correspond to formulas of AL, which is a restricted version of a multi-modal, multi-agent logic. Using the full logic (which we will call AL^+), both negative assumptions and *type-external* relationships can be represented. For instance, the following formula states that the FOPC content expression printed_on(userdoc,lw_plus) is not mutually believed:

$$\Box_{(B,S)} \neg \Box_{(B,M)} \text{printed\_on}(\text{userdoc}, \text{lw\_plus}) \tag{5.3}$$

A rule for inferring a belief assumption (SBUB) from a goal assumption (SBUW) is [Kobsa and Pohl, 1995]

$$\forall doc, p \, [\Box_{(B,S)}\Box_{(W,U)} \text{printed\_on}(doc, p) \rightarrow \Box_{(B,S)}\Box_{(B,U)} \text{printable\_on}(doc, p)] \tag{5.4}$$

General relationships between assumption types may be expressed by modal formula schemes. For example,

$$\Big(\Box_{(B,S)}\Box_{(W,U)} \Phi \wedge \Box_{(B,S)}\Box_{(B,U)} (\Phi \rightarrow \Psi)\Big) \rightarrow \Box_{(B,S)}\Box_{(W,U)} \Psi \tag{5.5}$$

(with Φ and Ψ being formula variables) represents the (meta-level) rule that users want any implication of their immediate goals if they know the implication relation.

Since AL^+ formulas and formula schemes provide the desired expressive power, such expressions shall be allowed in an *extended* AsTRa system. Therefore, we extend the definition of an AsTRa user modeling knowledge base:

Definition 5.19 (extended AsTRa knowledge base)
An extended AsTRa knowledge base is a tuple $\langle \mathcal{AT}, \mathcal{S}, \mathcal{MF}, \mathcal{MA} \rangle$ *where*

- $\mathcal{AT} = \{AT_1, \ldots, AT_n\}$ *is a set of assumption types,*
- $\mathcal{S} = \{S_1, \ldots, S_m\}$ *is a set of stereotypes,*
- $\mathcal{MF}$ *is a set of* AL^+ *formulas, and*
- $\mathcal{MA}$ *is a set of* AL^+ *formula schemes*

Because of the correspondence between type-internal UMKB contents and the AL^+ subset AL, all UMKB contents can be expressed in the modal logic AL^+.

More precisely, for an extended AsTRa UMKB, there is a corresponding set of AL^+ formulas, UMKB^{AL^+}, with

$$\mathrm{UMKB}^{AL^+} := (\bigcup_{AT \in \mathcal{AT}} \mathrm{KB}^{AL}_{AT}) \cup \mathcal{MF} \cup \mathcal{MA}$$

Hence, AsTRa reasoning could be completely based on modal logic reasoning functions for processing AL^+ knowledge bases, applied to UMKB^{AL^+}. However, with modal reasoning only, basic type-internal AsTRa reasoning with its flexibility of allowing several different formalisms for assumption contents would be abandoned. This is especially disadvantageous, if most parts of the UMKB can be represented within assumption types, possibly using specialized formalisms, and if most reasoning processes can be performed type-internally using the perhaps optimized facilities of content formalisms. If FOPC were the only content formalism of an AsTRa system, then type-internal AsTRa reasoning would come close to extended reasoning with the AL^+ subset AL. But still, FOPC formulas are less complex than the corresponding AL formulas.

In sum, there are enough reasons for requiring that an extended AsTRa implementation preserves assumption type knowledge bases. It implements modal reasoning as an additional reasoning level (which is however no longer type-internal). This reasoning level is specialized on AL^+ reasoning, and therefore is simply called AL^+.

For the AL^+ reasoning level, reasoning functions are needed that are applied to an AL^+ knowledge base, KB^{AL^+}, and an AL^+ expression ϕ. If such functions are available, they can get applied to the the AL^+ reformulation of the UMKB, UMKB^{AL^+}, and an AL^+-formulated user model content ϕ. Analogous to what the basic AsTRa framework requires of content formalisms, the extended framework requires the following functions to be available for AL^+ reasoning:

- AL^+-*derivable*$(\mathrm{KB}^{AL^+}, \phi)$
- AL^+-*forward*$(\mathrm{KB}^{AL^+}, \phi)$
- AL^+-*consistent*$(\mathrm{KB}^{AL^+}, \phi)$

With these functions, the means for realizing the AL^+ level are available.

Definition 5.20 (modal logic reasoning level AL^+)
Provided that general AL^+ reasoning functions are available as specified above, the reasoning functions of the AL^+ level can be defined as follows. Let ϕ be an AL^+ expression, then:

- $\mathbf{derivable}_{AL^+}(UMKB, \phi) := AL^+\text{-}derivable(UMKB^{AL^+}, \phi)$
- $\mathbf{forward}_{AL^+}(UMKB, \phi) := AL^+\text{-}forward(UMKB^{AL^+}, \phi)$

- **consistent**$_{AL^+}(UMKB, \phi) := AL^+\text{-}consistent(UMKB^{AL^+}, \phi)$

In an extended AsTRa system, not only a new reasoning level is available, but also the UMKB structure is extended in comparison to a basic system. Therefore, the "database functions" **store** and **fetch** need to be redefined, too. In basic AsTRa, $store_F$ and $fetch_F$ functions of content formalisms are sufficient to handle type-internal expressions. For extended AsTRa, three additional cases need to be distinguished: If an input formula ϕ is an AL formula $\mathcal{M}(AT)\,ac$, which corresponds to a type-internal expression $AT{:}ac$, it will be handled by type-internal mechanisms (e.g., of level 0); otherwise, it will be added to or retrieved from the sets $\mathcal{MF}$ or $\mathcal{MA}$, depending on its being a modal formula or a modal axiom, respectively.

- **store**$(\text{UMKB}, \phi) := \begin{cases} store_{F(ac)}(\text{KB}_{AT}, ac) & \phi = AT{:}ac \\ store_{F(ac)}(\text{KB}_{AT}, ac) & \phi = \mathcal{M}(AT)\,ac \in AL \\ \mathcal{MF} := \mathcal{MF} \cup \{\phi\} & \phi \text{ is an } AL^+ \text{ formula } \notin AL \\ \mathcal{MA} := \mathcal{MA} \cup \{\phi\} & \phi \text{ is an } AL^+ \text{ axiom} \end{cases}$

- **fetch**$(\text{UMKB}, \phi) := \begin{cases} fetch_{F(ac)}(\text{KB}_{AT}, ac) & \phi = AT{:}ac \\ fetch_{F(ac)}(\text{KB}_{AT}, ac) & \phi = \mathcal{M}(AT)\,ac \in AL \\ \phi \in \mathcal{MF} & \phi \text{ is an } AL^+ \text{ formula } \notin AL \\ \phi \in \mathcal{MA} & \phi \text{ is an } AL^+ \text{ axiom} \end{cases}$

With the availability of the AL^+ reasoning level, an extended AsTRa implementation can offer AL^+ reasoning *as an additional option.* So, full expressive power becomes available, but the flexibility of AsTRa is not sacrificed.

In order to make its internal mechanisms transparent to the user model developer, an extended AsTRa implementation may admit AL^+ as a uniform interface language for UMKB contents, thus permitting top-down access to the UMKB as an alternative to standard bottom-up access of the basic framework. This requires that simple AL expressions of the form $\mathcal{M}(AT)\,ac$ can be detected among AL^+ inputs and, if desired, can be handled like their corresponding basic AsTRa expressions $AT{:}ac$. To do so, the appropriate content formalism $F(ac)$ must be determined. This can be done by looking for the formalism F such that $ac \in t_F(\mathcal{L}_F)$, i.e., ac is among the FOPC translations of F-expressions. The problem may occur that the FOPC part ac of $\mathcal{M}(AT)\,ac$ cannot be related to a content formalism of the AsTRa system. There can be two reasons for this: First, there may be no F such that $ac \in t_F(\mathcal{L}_F)$. Second, there may be more than one formalism that satisfies the above condition. For both cases, there are remedies: In the first case, $\mathcal{M}(AT)\,ac$ is handled as the AL^+ expression that it is. In the second case, a total preference order of content formalisms is needed; then, ac will be handled by the formalism that is preferred according to this order.

Figure 5.2 illustrates the above ideas and shows the alternative access possibilities to an AsTRa UMKB. Basic AsTRa facilities are typically accessed bottom-up

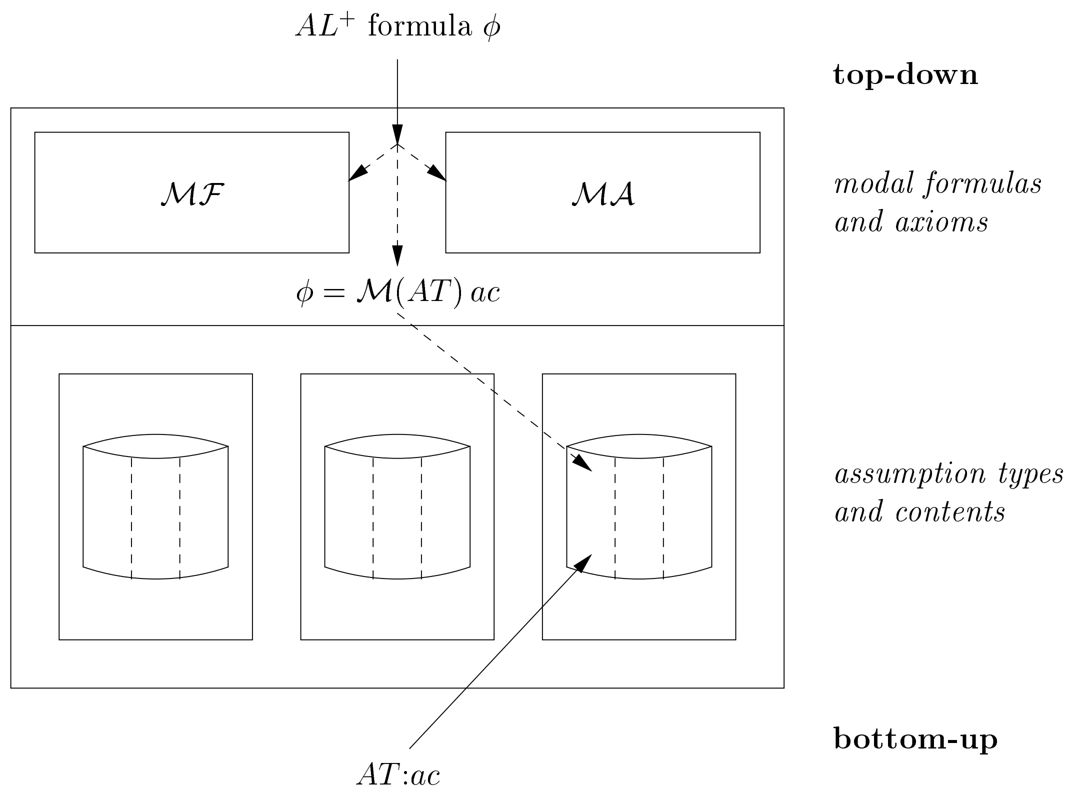


Figure 5.2: Bottom-up and top-down access to an AsTRa UMKB

via assumption type expressions $AT{:}ac$. This way, assumption type and content formalism can be chosen directly. Alternatively, an AsTRa UMKB can be accessed top-down via AL^+ formulas ϕ. This is the only way, if type-external formulas are dealt with. An AsTRa system will determine whether ϕ is a modal axiom, a complex modal formula, or an AL formula $\mathcal{M}(AT)\,ac$. In the latter case, it will determine the appropriate assumption type and content formalism automatically.

5.3.2 On the Role of Type-Internal and Type-External User Modeling Knowledge

With the exception of those SB assumption contents that are not related to the user (e.g., background knowledge about the domain), type-internal knowledge represents assumptions that the user modeling system has about the user. This knowledge is mostly acquired from the interaction between user and application system during the *run time* phase of the user modeling process. On the contrary, stereotyped assumptions are normally pre-defined during the *development time* of a user modeling system, but they become type-internal knowledge at run time, too, when a user is assigned to a user group. Type-internal reasoning is reasoning with type-internal knowledge, and hence relies on inference rules that are ascribed to the user. Therefore, this kind of reasoning can be regarded as a simulation of the user's reasoning.

Type-external knowledge plays a different role. Modal formula schemes can be regarded as axioms that influence the properties of the whole type-external reasoning process, but they can also be regarded as (general) inference rules of the system. Non-schematic type-external implications represent specific inference rules that the system may apply to acquire implicit assumptions from the current user model. A somewhat mixed case is reasoning with inference rules of the system that concern one assumption type only. It is one of the basic properties of modal logic that rules within the scope of an operator can do neither more nor less than rules with the same operator on both sides of the implication. This is simply due to modal modus ponens, which is a sound inference rule for every modal logic with a possible worlds semantics (cf. Section 3.4.1). For instance, the type-external system rule

$$(\Box_{(B,S)}\Box_{(B,U)}\text{printable-on}(doc, pr1)) \rightarrow (\Box_{(B,S)}\Box_{(B,U)}\text{printable-on}(doc, pr2))$$

does not allow more conclusions than the SBUB-internal rule

$$\text{printable-on}(doc, pr1) \rightarrow \text{printable-on}(doc, pr2)$$

which corresponds to the AL expression

$$\Box_{(B,S)}\Box_{(B,U)}(\text{printable-on}(doc, pr1) \rightarrow \text{printable-on}(doc, pr2))$$

In an AsTRa system, the type-external rule will probably lead to fewer conclusions, since it is not applied in type-internal reasoning. It is in the hand and responsibility of the user model developer to decide, which kind of representation is most appropriate for the rules that are supposed to describe possible non-simulative system inferences concerning only one assumption type.

There is a second problematic case, namely negative assumptions. In Section 2.2, assumptions about what a user does not believe or want, or other assumptions that involve negated modalities, were regarded as distinct kinds of assumptions. In most cases, they represent assumptions about the user. In this respect, they are more similar to type-internal assumptions than to type-external assumptions. In the following section, we will therefore explore how type-internal representation and reasoning can be extended to cover negative assumptions. It will also become clear why negative assumptions do not fit into the basic AsTRa framework.

5.4 Negative Assumption Types

5.4.1 Introduction

In Section 5.3.1, we showed that negative assumptions can be represented as formulas of the modal logic AL^+. However, for several reasons it is worth taking a closer look at them. These formulas differ from simple AL formulas, which correspond to type-internal knowledge, only in the negation operator that may appear in front of modal operators. Above, we discussed that the role of negative assumptions is quite similar to that of type-internal knowledge in an AsTRa system. Furthermore, negative assumptions constitute an interesting subclass of user modeling knowledge. Several user modeling systems made use of negative assumptions without needing any other type-external representation or reasoning methods (e.g., see [Huang *et al.*, 1991; Kobsa *et al.*, 1994]). Hence, it makes sense to deal with negative assumptions in a specialized way. Because of the similarity between negative assumptions and type-internal knowledge, such a method is likely to be closely related to type-internal mechanisms. A special treatment will probably turn out to be more efficient than the quite general approach of modal reasoning, especially if negative assumptions are the only kind of knowledge to be represented in a UMKB that cannot be handled with basic AsTRa reasoning.

Like the positive assumptions that a basic AsTRa system can handle, negative assumptions can be construed as belonging to different types. In Section 2.2, a notation for negative assumption types was introduced: The negation symbol ∼ can be used for labels of negative assumption types. For example, SB∼UB can denote the assumption type of all system beliefs about what the user does not believe. Analogously, other negative assumption types are possible, e.g.:

SB∼UW: assumptions about what is not a user goal

SB∼MBUB: assumptions about what is not mutually believed about user beliefs

SBUBSB∼UB: assumptions about the user's beliefs about what the system believes her to not believe

Note that an assumption content p of, e.g., type SB∼UB is not equivalent to an assumption content $\neg p$ of type SBUB. The first case represents a negative assessment of S concerning a belief of U, while the second case represents a negative assessment of U concerning a fact denoted by p. However, there are relationships between such differently negated expressions. In the next section, we will develop inference mechanisms that, based on modal logic semantics, exploit these relationships.

As with standard assumption types, negative assumption type labels start with an 'SB' agent-modality pair. A negation '∼SB' is not allowed at the beginning of a label. It does not seem to make sense to store in a UMKB, i.e. in the collection of the system's assumptions about the user and the world, something that the system does not assume and hence should be no part of the UMKB. All following agent-modality pairs may be negated; and in contrast to the examples given above, there may be arbitrarily many negations in an assumption type label.

The following definition redefines the notion of an assumption type in order to additionally cover negative assumption types. The main aspect of an assumption type remains unchanged: the associated knowledge base.

Definition 5.21 ((negative) assumption type) *An assumption type is a pair* $\langle AT, KB_{AT} \rangle$ *where*

1. $AT = \nu_1 a_1 m_1 \ldots \nu_n a_n m_n$ *is the assumption type label. AT is a sequence of triples of negations* ν_i*, agents* a_i*, and modalities* m_i*, which must satisfy several conditions:* $n \geq 1$*,* $\nu_i = \sim$ *or* $\nu_i = \varepsilon$[5]*,* $a_i \in \{S, U, M\}$*,* $m_i \in \mathcal{MOD}$ *with* $\{B\} \subseteq \mathcal{MOD}$*, and* $a_i = M$ *only if* $m_i = B$*. An assumption type label starts with SB, i.e.* $\nu_1 = \varepsilon$*,* $a_1 = S$*, and* $m_1 = B$*.*

2. KB_{AT} *is the assumption type knowledge base. On an abstract level, it can be seen as a set of expressions of knowledge representation formalisms.*

It was quite easy to syntactically introduce the concept of negative assumption types similar to that of standard "positive" assumption types. However, when (type-internal) reasoning is concerned, there are fundamental differences between negative and positive types, as we will further discuss in the following subsection.

Recall that negative assumption types were introduced to replace modal AL^+ formulas of a specific form in extended AsTRa. I.e., also for negative assumption

[5] Here, ε denotes the empty word or empty string.

types, every type-internal expression $AT{:}ac$ has a corresponding modal formula $\mathcal{M}(AT)\,ac$. For an assumption type $AT = \nu_1 a_1 m_1 \ldots \nu_n a_n m_n$, the modal operator sequence $\mathcal{M}(AT)$ may now contain negations, i.e.

$$\mathcal{M}(AT) = \mathrm{sig}_1 \Box_{(m_1,a_1)} \ldots \mathrm{sig}_n \Box_{(m_n,a_n)} \tag{5.6}$$

where $\mathrm{sig}_i = \neg$ if $\nu_i = \sim$, and $\mathrm{sig}_i = \nu_i = \varepsilon$ else. The resulting extension of AL that permits negated modal operators, will further be called $AL^{\neg}$; obviously, $AL^{\neg}$ is a subset of AL^{+}. E.g., the type-internal expression SB∼UB:PSprinter(lj1) corresponds to the $AL^{\neg}$ formula $\Box_{(B,S)}\neg\Box_{(B,U)}\mathrm{PSprinter(lj1)}$.

5.4.2 Type-Internal Reasoning Is Not Sound

For positive assumption types, it could be shown that type-internal reasoning is equivalent to reasoning in the limited modal logic AL. The fundamental property of basic AsTRa is that, if an assumption type knowledge base entails a content expression, $\mathrm{KB}_{AT} \models ac$, an entailment will also hold for the corresponding AL expressions: $\mathrm{KB}_{AT}^{AL} \models \mathcal{M}(AT)\,ac$. For negative assumption types and $AL^{\neg}$, this *entailment correspondence* does not exist.

Intuitively, it is quite clear that, if a user is assumed to not believe a proposition a and to not believe that a implies b (i.e., if a and $a \rightarrow b$ are contents of type SB∼UB), it is implausible to derive that she is also assumed to not believe in b (i.e., that also b is a content of SB∼UB). However, not only intuition, but also modal logic semantics of negative assumptions make that clear. In the following, the modal "diamond" operator $\Diamond$ will often be used in addition to the "box" operator $\Box$. Recall that $\Diamond\phi$ holds in a world w if there is a world accessible from w where ϕ holds (cf. Definition 3.8); hence, $\Diamond\phi$ is an abbreviation for $\neg\Box\neg\phi$. Then, the negation of an AL^{+} operator, $\neg\Box_{(m,a)}\phi$, is equivalent to $\Diamond_{(m,a)}\neg\phi$, i.e., there is an accessible world where $\neg\phi$ holds. Now, a (type-internal) entailment $\phi \models \psi$ does not help to infer that there is also an accessible world where $\neg\psi$ holds, i.e. that $\Diamond_{(m,a)}\neg\psi$ holds, which is equivalent to $\neg\Box_{(m,a)}\psi$. In brief, $\phi \models \psi$ is *not* equivalent to $\neg\Box_{(m,a)}\phi \models \neg\Box_{(m,a)}\psi$ (while it *is* equivalent to $\Box_{(m,a)}\phi \models \Box_{(m,a)}\psi$, which is the formal background of the above-mentioned entailment correspondence in case of positive assumption types).

In order to discuss negative assumptions more deeply, we introduce a special *box-diamond form* for $AL^{\neg}$ formulas, which permits only non-negated $\Box$ and $\Diamond$ operators in the modal operator sequence $\mathcal{M}(AT)$. This form is generated from the standard form for negative assumptions with negated and non-negated $\Box$ operators by moving negations into the scope of modal operators. The transformation is possible because of the above-mentioned equivalence

$$\neg\Box\phi \equiv \Diamond\neg\phi$$

and the elimination rule for double negations

$$\neg\neg\Box\phi \equiv \Box\phi$$

Definition 5.22 (box-diamond form)

- *The* box-diamond form *of an $AL^{\neg}$ formula (let $sig_i \in \{\neg, \varepsilon\}$)*

$$\mathcal{M}(AT)\, ac \;=\; sig_1 \Box_{(m_1,a_1)} \ldots sig_n \Box_{(m_n,a_n)}\, ac$$

  is a modal formula

$$O_{(m_1,a_1)} \ldots O_{(m_n,a_n)}\, ac'$$

  where $O \in \{\Box, \Diamond\}$ and $ac' \in \{ac, \neg ac\}$. The box-diamond form is constructed by applying the following two transformations from left to right:

  1. $\neg\neg\Box\phi \;\mapsto\; \Box\phi$
  2. $\neg\Box\phi \;\mapsto\; \Diamond\neg\phi$

- *AT determines the $O_{(m_i,a_i)}$ and the sign of ac'. So, for every assumption type AT, we define a pair BD(AT) := $\langle$bds(AT),sig(AT)$\rangle$, where*
  - *bds(AT) := $O_{(m_1,a_1)} \ldots O_{(m_n,a_n)}$ is the* box-diamond sequence *of AT, and*
  - *sig(AT) is defined as the negation symbol $\neg$ if $ac' = \neg ac$, and as the empty symbol ε else.*

- *For a pair $\langle bds, sig\rangle$ where bds is a box-diamond sequence and $sig \in \{\neg, \varepsilon\}$, we define $BD^{-1}(\langle bds, sig\rangle) := \langle AT, sig'\rangle$ with $\mathcal{M}(AT)\, sig'\, \phi$ being the equivalent $AL^{\neg}$ formula to bds sig ϕ (formally: $\mathcal{M}(AT)\, sig'\, \phi \;\in AL^{\neg}$ and $\mathcal{M}(AT)\, sig'\, \phi \;\equiv\;$ bds sig ϕ).*

- *We define $\mathcal{BDS}$ as the set of all possible box-diamond sequences, i.e.*

$$\mathcal{BDS} := \{BD(AT) \mid AT \textit{ is a positive or negative assumption type}\}$$

Note that AL expressions, which represent positive assumptions, are already in box-diamond form, since their modal operator sequence only contains non-negated $\Box$ operators.

Example 5.1:

1. Since $\Box_{(B,S)}\neg\Box_{(W,U)}\phi \;\mapsto\; \Box_{(B,S)}\Diamond_{(W,U)}\neg\phi$,

$$\begin{aligned}
&\mathrm{BD(SB{\sim}UW)} = \langle \Box_{(B,S)}\Diamond_{(W,U)}, \neg\rangle \\
&\mathrm{BD}^{-1}(\langle \Box_{(B,S)}\Diamond_{(W,U)}, \neg\rangle = \langle \mathrm{SBUW}, \varepsilon\rangle \\
&\mathrm{BD}^{-1}(\langle \Box_{(B,S)}\Diamond_{(W,U)}, \varepsilon\rangle = \langle \mathrm{SBUW}, \neg\rangle
\end{aligned}$$

2. Since $\Box_{(B,S)}\Box_{(B,U)}\neg\Box_{(B,S)}\neg\Box_{(B,U)}\phi \;\mapsto\; \Box_{(B,S)}\Box_{(B,U}\Diamond_{(B,S)}\Box_{(B,U)}\phi$,

$$\mathrm{BD}(\mathrm{SBUB{\sim}SB{\sim}UB}) = \langle \Box_{(B,S)}\Box_{(B,U}\Diamond_{(B,S)}\Box_{(B,U)}, \varepsilon\rangle$$
$$\mathrm{BD}^{-1}(\langle \Box_{(B,S)}\Box_{(B,U}\Diamond_{(B,S)}\Box_{(B,U)}, \varepsilon\rangle) = \langle \mathrm{SBUB} \sim \mathrm{SB} \sim \mathrm{UB}, \varepsilon\rangle$$
$$\mathrm{BD}^{-1}(\langle \Box_{(B,S)}\Box_{(B,U}\Diamond_{(B,S)}\Box_{(B,U)}, \neg\rangle) = \langle \mathrm{SBUB} \sim \mathrm{SB} \sim \mathrm{UB}, \neg\rangle$$

In Section 5.2.6, it was stated that a modal operator sequence $\mathcal{M}(AT)$ of a (positive) AL formula—and, consequently, also the corresponding assumption type AT —determines a set of AT-accessible worlds. A (positive) AL formula is satisfied by a modal interpretation if and only if the content formula ac is satisfied by the interpretation in all AT-accessible worlds. If all AL formulas of one type, i.e. the set KB_{AT}^{AL}, are satisfied, then KB_{AT} is satisfied in all AT-accessible worlds. If KB_{AT} entails a formula ac, then also ac is satisfied in all AT-accessible worlds, and hence $\mathcal{M}(AT)\,ac$ is satisfied. So, $\mathrm{KB}_{AT}^{AL} \models \mathcal{M}(AT)\,ac$ if $\mathrm{KB}_{AT} \models ac$.

The box-diamond form allows to clearly show that this argumentation does not hold for negative assumptions. Also the interpretation of a $AL^{\neg}$ formula depends on the interpretation of a content expression in a possible world. But if a $\Diamond$ operator is contained in the operator sequence of its box-diamond form, a selection is made from the set of accessible worlds, since the $\Diamond$ operator only requires the existence of one accessible world where the following sub-formula is true. That is, an $AL^{\neg}$ formula $\phi = \mathcal{M}(AT)\,ac$ is satisfied if there is some subset of the worlds that are accessible according to the operator sequence of its box-diamond form $BD(\phi) = bds(AT)\,ac'$, where ac' is satisfied. Since the selection that is made for a $\Diamond$ operator depends on the following sub-formula, for two different $AL^{\neg}$ formulas the satisfying subsets may be different. Hence, the above statement: "If all formulas of one type, i.e. KB_{AT}^{AL}, are satisfied, then KB_{AT} is satisfied in all AT-accessible worlds" cannot be repeated here. In case $ac' = \neg ac$, not the content expressions $ac \in \mathrm{KB}_{AT}$ are satisfied in the accessible worlds, but their negations. In case $ac' = ac$, KB_{AT} is not satisfied in all accessible worlds, but its elements are satisfied in possibly different subsets of the accessible worlds.

In sum, type-internal inferences within a negative assumption type AT are not sound with respect to the semantics of the corresponding modal logic knowledge base $\mathrm{KB}_{AT}^{AL^{\neg}}$. Therefore, for an AsTRa system with negative assumption types, the type-internal reasoning functions $\mathbf{derivable}_{TI}$ and $\mathbf{forward}_{TI}$ must be redefined. For negative assumption types, the $derivable_F$ and $forward_F$ functions of the concerned content formalisms cannot be called, since they will draw unsound type-internal inferences. Instead, only the database functions $fetch_F$ and $store_F$ can be invoked. As a consequence, we revise Definition 5.15 by allowing for negative assumption types AT:

$$\mathbf{derivable}_{TI}(\mathrm{UMKB}, AT{:}ac) := \begin{cases} derivable_{F(ac)}(\mathrm{KB}_{AT}, ac) & AT \text{ positive} \\ fetch_{F(ac)}(\mathrm{KB}_{AT}, ac) & AT \text{ negative} \end{cases}$$

$$\mathbf{forward}_{TI}(\mathrm{UMKB}, AT{:}ac) := \begin{cases} forward_{F(ac)}(\mathrm{KB}_{AT}, ac) & AT \text{ positive} \\ store_{F(ac)}(\mathrm{KB}_{AT}, ac) & AT \text{ negative} \end{cases}$$

Hence, for negative assumption types, reasoning level TI is equal to reasoning level 0. A problem occurs, if $F(ac)$ is a simple content formalism, because then, **store** and **fetch** will employ the simple $tell_{F(ac)}$ and $ask_{F(ac)}$ access functions. Therefore, in an AsTRa implementation, simple content formalisms F must take care of negative assumption types; both $tell_F$ and ask_F must not do reasoning if applied to the knowledge base of a negative assumption type.

Also the $\mathbf{consistent}_{TI}$ function needs to be redefined for negative assumption types. For a type-internal expression $AT{:}ac$, the content expression ac may be inconsistent with the assumption type knowledge base KB_{AT}. In case of positive assumption types, this is equivalent with the AL expression $\mathcal{M}(AT)\,ac$ being inconsistent with UMKB^{AL}. In case of a negative assumption type AT, in general two $AL^{\neg}$ formulas $\mathcal{M}(AT)\,ac_1$ and $\mathcal{M}(AT)\,ac_2$ are not inconsistent, even if ac_1 and ac_2 are contradictory. Both formulas can be satisfied, if the non-modal parts of their box-diamond forms, ac_1' and ac_2', are satisfied in disjoint subsets of the accessible worlds. This means that type-internal consistency checks are not sound in case of negative assumption types. So, also $\mathbf{consistent}_{TI}$ must be reduced to level 0, if applied to a negative type:

$$\mathbf{consistent}_{TI}(\mathrm{UMKB}, AT{:}ac) := \begin{cases} consistent_{F(ac)}(\mathrm{KB}_{AT}, ac) & AT \text{ positive} \\ \mathbf{true} & AT \text{ negative} \end{cases}$$

5.4.3 Specialized Reasoning for Negative Types

On the one hand, the introduction of negative assumption types has several advantages. It allows one to avoid the use of possibly expensive modal reasoning techniques for modal expressions of a still quite restricted form. Furthermore, content formalisms may now be used for the representation of both positive and negative assumption contents. On the other hand, the restriction of reasoning about negative assumptions to reasoning with negative assumption type knowledge bases leads to a primitive store and fetch interface. Real inferences with negative assumptions are not possible.

In Section 2.2, negative assumptions were characterized as logic-based graduations of positive assumptions. This observation adds to the image that negative assumptions only make sense in the context of related positive or perhaps other negative assumptions. One of the basic axioms of modal logic is the axiom D

$$\Box\phi \;\rightarrow\; \neg\Box\neg\phi \qquad \text{(D)}$$

which holds for an operator, if the corresponding accessibility relation is serial[6]. Using the diamond operator, the axiom can be written as

$$\Box\phi \rightarrow \Diamond\phi$$

[6] An accessibility relation is serial, if for all words there is at least one successor world, formally: $\forall w \exists w'\, R(w, w')$.

D relates basic positive and negative assumptions. With the help of the necessitation rule, axiom K, and modus ponens, it can be derived that the implication of D remains valid if both antecedent and conclusion are within the scope of another $\Box$ operator:

$$\Box\Box\phi \rightarrow \Box\Diamond\phi \tag{5.7}$$

The contraposition of this implication is equivalent to

$$\Diamond\Box\neg\phi \rightarrow \Diamond\Diamond\neg\phi$$

so that we arrive at another axiomatic implication (with $\psi := \neg\phi$):

$$\Diamond\Box\psi \rightarrow \Diamond\Diamond\psi$$

Furthermore, an instantiation of axiom D with $\phi := \Diamond\phi$ leads to

$$\Box\Diamond\phi \rightarrow \Diamond\Diamond\phi \tag{5.8}$$

Finally, the combination of formulas (5.7) and (5.8) results in

$$\Box\Box\phi \rightarrow \Diamond\Diamond\phi$$

The common property of all these implications is that the conclusion is obtained by replacing at least one $\Box$ operator of the antecedent with a $\Diamond$ operator. Since the above reasoning steps can be performed for box-diamond sequences of any length, we obtain the following corollary from D:

Corollary 3 *If all accessibility relations $R_{(m,a)}$ in a possible-worlds frame for $AL^{\neg}$ are serial (i.e., $\Box_{(m,a)}\phi \rightarrow \Diamond_{(m,a)}\phi$ holds for all m, a) then*

$$O_{1(m_1,a_1)} \ldots O_{n(m_n,a_n)}\phi \;\rightarrow\; O'_{1(m_1,a_1)} \ldots O'_{n(m_n,a_n)}\phi$$

holds, where $O'_i = \Diamond$ if $O_i = \Diamond$, and $O'_i \in \{\Box, \Diamond\}$ if $O_i = \Box$.

That is, axiom D entails a family of implications, which can be used for UMKB derivations concerning negative assumption types. From Corollary 3, we immediately arrive at the following statement about entailment between expressions in box-diamond form:

Corollary 4

$$O_{1(m_1,a_1)} \ldots O_{n(m_n,a_n)}\phi \models \textit{(entails)}\; O'_{1(m_1,a_1)} \ldots O'_{n(m_n,a_n)}\phi$$

where $O'_i = \Diamond$ if $O_i = \Diamond$, and $O'_i \in \{\Box, \Diamond\}$ if $O_i = \Box$.

An immediate application of that is forward reasoning. We define a relation FOR between box-diamond sequences:

Definition 5.23 *FOR is a relation over the set $\mathcal{BDS}$ of box-diamond sequences so that for two sequences $bds_1 \in \mathcal{BDS}$ and $bds_2 \in \mathcal{BDS}$*

$$FOR(bds_1, bds_2) \iff bds_1\phi \models \text{(entails) } bds_2\phi \text{ and } bds_1 \neq bds_2$$

for all content formulas ϕ.

The condition $bds_1 \neq bds_2$ is to exclude the trivial case that a formula entails itself.

Example 5.2:

1. $\Box_{(B,S)}\Box_{(W,U)}\phi$ entails $\Box_{(B,S)}\Diamond_{(W,U)}\phi$, $\Diamond_{(B,S)}\Box_{(W,U)}\phi$, $\Box_{(B,S)}\Diamond_{(W,U)}\phi$, and itself. Therefore,

$$\begin{array}{l} \mathrm{FOR}(\Box_{(B,S)}\Box_{(W,U)}, \Box_{(B,S)}\Diamond_{(W,U)}) \\ \mathrm{FOR}(\Box_{(B,S)}\Box_{(W,U)}, \Diamond_{(B,S)}\Box_{(W,U)}) \\ \mathrm{FOR}(\Box_{(B,S)}\Box_{(W,U)}, \Diamond_{(B,S)}\Diamond_{(W,U)}) \end{array}$$

2. $\Box_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}\phi$ entails $\Diamond_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}\phi$ and itself. Therefore,

$$\mathrm{FOR}(\Box_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}, \Diamond_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)})$$

The result of Corollary 3 is also relevant in case of queries to the UMKB. For a query that concerns a (negative) assumption type, it is crucial to know by which other formula the queried formula is entailed. Therefore, a kind of inverse of Corollary 4 is needed. Note that we continue to presume axiom D.

Corollary 5

$$O_{1(m_1,a_1)} \dots O_{n(m_n,a_n)}\phi$$

is entailed by all formulas

$$O'_{1(m_1,a_1)} \dots O'_{n(m_n,a_n)}\phi$$

where $O'_i = \Box$ if $O_i = \Box$, and $O'_i \in \{\Box, \Diamond\}$ if $O_i = \Diamond$.

This result is employed to define a backward derivability relation DER between box-diamond sequences:

Definition 5.24 *DER is a relation over the set $\mathcal{BDS}$ of box-diamond sequences so that for two sequences $bds_1 \in \mathcal{BDS}$ and $bds_2 \in \mathcal{BDS}$*

$$DER(bds_1, bds_2) \iff bds_1\phi \text{ is entailed by } bds_2\phi \text{ and } bds_1 \neq bds_2$$

for all content formulas ϕ.

The condition $bds_1 \neq bds_2$ is to exclude the trivial case that a formula is entailed by itself.

Example 5.3:

1. $\Box_{(B,S)}\Diamond_{(W,U)}\phi$ is entailed by $\Box_{(B,S)}\Box_{(W,U)}\phi$ and by itself. Therefore,

   $$\mathrm{DER}(\Box_{(B,S)}\Diamond_{(W,U)}, \Box_{(B,S)}\Box_{(W,U)})$$

2. $\Box_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}\phi$ is entailed by $\Box_{(B,S)}\Box_{(B,M)}\Diamond_{(B,U)}\phi$, $\Box_{(B,S)}\Diamond_{(B,M)}\Box_{(B,U)}\phi$, $\Box_{(B,S)}\Box_{(B,M)}\Box_{(B,U)}\phi$, and by itself. Therefore,

   $$\mathrm{DER}(\Box_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}, \Box_{(B,S)}\Box_{(B,M)}\Diamond_{(B,U)})$$
   $$\mathrm{DER}(\Box_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}, \Box_{(B,S)}\Diamond_{(B,M)}\Box_{(B,U)})$$
   $$\mathrm{DER}(\Box_{(B,S)}\Diamond_{(B,M)}\Diamond_{(B,U)}, \Box_{(B,S)}\Box_{(B,M)}\Box_{(B,U)})$$

In Section 5.4.2, we mentioned two fundamental obstacles to the application of standard type-internal reasoning procedures to negative assumption types: Both type-internal derivations and consistency checks are not sound in case of negative types. As far as derivations are concerned, Corollary 5 has shown that axiom D provides opportunities that can be taken. But also inconsistency relationships between positive and negative assumptions exist, which we will again discuss with the help of the box-diamond form of $AL^{\neg}$ formulas. Note that

$$\neg\Box\phi \equiv \Diamond\neg\phi$$

so that $\Box\phi$ is inconsistent with $\Diamond\neg\phi$ and vice versa. Furthermore,

$$\neg\Box\neg\phi \equiv \Diamond\phi$$

so that $\Box\neg\phi$ is inconsistent with $\Diamond\phi$ and vice versa.

In general, the negation of a box-diamond form is a box-diamond form with switched operators and negated inner formula:

Lemma 2

$$\neg(O_{1(m_1,a_1)}\ldots O_{n(m_n,a_n)}\phi) \;\equiv\; O'_{1(m_1,a_1)}\ldots O'_{n(m_n,a_n)}\neg\phi$$

where $O'_i = \Diamond$ *if* $O_i = \Box$, *and* $O'_i = \Box$ *if* $O_i = \Diamond$.

But an $AL^{\neg}$ formula is not only inconsistent with its own negation, but also with the negation of all its entailments. So, from the above Lemma and Corollary 3, we can make a statement about inconsistencies, which can also be regarded as a corollary from axiom D.

Corollary 6 $O_{1(m_1,a_1)}\ldots O_{n(m_n,a_n)}\phi$ *is inconsistent with* $O'_{1(m_1,a_1)}\ldots O'_{n(m_n,a_n)}\neg\phi$ *where* $O'_i = \Box$ *if* $O_i = \Diamond$, *and* $O_i \in \{\Box, \Diamond\}$ *if* $O_i = \Box$.

In analogy to relations FOR and DER, a relation INC between box-diamond sequences is defined:

Definition 5.25 *INC is a relation over the set $\mathcal{BDS}$ of box-diamond sequences so that for two sequences $bds_1 \in \mathcal{BDS}$ and $bds_2 \in \mathcal{BDS}$*

$$INC(bds_1, bds_2) \iff bds_1\phi \text{ is inconsistent with } bds_2\neg\phi \text{ and } bds_1 \neq bds_2$$

for all content formulas ϕ.

The condition $bds_1 \neq bds_2$ is to exclude the trivial case of the type-internal inconsistency of an assumption content ac with its negation, which may occur in case of a positive assumption type.

Example 5.4:

1. $\Box_{(B,S)}\Diamond_{(W,U)}\phi$ is inconsistent with $\Box_{(B,S)}\Box_{(W,U)}\neg\phi$ and with $\Diamond_{(B,S)}\Box_{(W,U)}\neg\phi$. Hence,

   $\mathrm{INC}(\Box_{(B,S)}\Diamond_{(W,U)}, \Box_{(B,S)}\Box_{(W,U)})$ and
   $\mathrm{INC}(\Box_{(B,S)}\Diamond_{(W,U)}, \Diamond_{(B,S)}\Box_{(W,U)})$

2. $\Box_{(B,S)}\Box_{(W,U)}\phi$ is inconsistent with $\Box_{(B,S)}\Diamond_{(W,U)}\neg\phi$, $\Diamond_{(B,S)}\Box_{(W,U)}\neg\phi$, $\Diamond_{(B,S)}\Diamond_{(W,U)}\neg\phi$, and $\Box_{(B,S)}\Box_{(W,U)}\neg\phi$. Hence,

   $\mathrm{INC}(\Box_{(B,S)}\Box_{(W,U)}, \Box_{(B,S)}\Diamond_{(W,U)})$,
   $\mathrm{INC}(\Box_{(B,S)}\Box_{(W,U)}, \Diamond_{(B,S)}\Box_{(W,U)})$, and
   $\mathrm{INC}(\Box_{(B,S)}\Box_{(W,U)}, \Diamond_{(B,S)}\Diamond_{(W,U)})$

Now, the relations FOR, DER, and INC can be employed to establish forward derivation, backward derivation, and inconsistency relations, respectively, between assumption types. The following principle is applied: An $AL^{\neg}$ formula ϕ is equivalent to its box-diamond form ϕ'. ϕ' entails (can be derived from, is inconsistent with) a number of box-diamond forms ψ_i' according to FOR (DER, INC), which are equivalent to $AL^{\neg}$ formulas ψ_i. Hence, ϕ entails (can be derived from, is inconsistent with) all ψ_i. More formally, for an assumption type AT we define forward(AT), derivable(AT), and inconsistent(AT) as sets of pairs $\langle AT', sig\rangle$ such that $\mathcal{M}(AT)\,ac$ entails (can be derived from, is inconsistent with) $\mathcal{M}(AT')\,sig\,ac$. sig can be $\neg$ or ε; it indicates if ac needs to be negated or not.

Definition 5.26 (forward(AT), derivable(AT), inconsistent(AT)) *For an assumption type AT, the sets forward(AT), derivable(AT) and inconsistent(AT) are defined as follows (with $bds, bds' \in \mathcal{BDS}$, and $sig \in \{\neg, \varepsilon\}$):*

1. *forward(AT) :=*
   $\{BD^{-1}(\langle bds', sig\rangle) \mid BD(AT) = \langle bds, sig\rangle \text{ and } FOR(bds, bds')\}$

2. *derivable(AT) :=*
   $\{BD^{-1}(\langle bds', sig\rangle) \mid BD(AT) = \langle bds, sig\rangle$ *and* $DER(bds, bds')\}$

3. *inconsistent(AT) :=*
   $\{BD^{-1}(\langle bds', \overline{sig}\rangle) \mid BD(AT) = \langle bds, sig\rangle$ *and* $INC(bds, bds')\}$

$\overline{sig}$ *denotes the inverse of sig,* ($\overline{\neg} = \varepsilon$ *and* $\overline{\varepsilon} = \neg$)*.* *forward*$(AT)$*,* *derivable*$(AT)$*, and inconsistent*$(AT)$ *may contain pairs* $\langle AT', sig\rangle$ *where* AT' *does not start with SB. Such pairs are removed.*

Example 5.5:

1. BD(SB∼UW) = $\langle \Box_{(B,S)}\Diamond_{(W,U)}, \neg\rangle$

   (a) FOR($\Box_{(B,S)}\Diamond_{(W,U)}, \Diamond_{(B,S)}\Diamond_{(W,U)}$)
   $\text{BD}^{-1}(\langle \Diamond_{(B,S)}\Diamond_{(W,U)}, \neg\rangle = \langle\sim\text{SBUW}, \varepsilon\rangle$
   $\Longrightarrow$ forward(SB∼UW) = $\emptyset$

   (b) DER($\Box_{(B,S)}\Diamond_{(W,U)}, \Box_{(B,S)}\Box_{(W,U)}$)
   $\text{BD}^{-1}(\langle \Box_{(B,S)}\Box_{(W,U)}, \neg\rangle = \langle\text{SBUW}, \neg\rangle$
   $\Longrightarrow$ derivable(SB∼UW) = $\{\langle\text{SBUW}, \neg\rangle\}$

   (c) INC($\Box_{(B,S)}\Diamond_{(W,U)}, \Box_{(B,S)}\Box_{(W,U)}$)
   INC($\Box_{(B,S)}\Diamond_{(W,U)}, \Diamond_{(B,S)}\Box_{(W,U)}$)
   $\text{BD}^{-1}(\langle \Box_{(B,S)}\Box_{(W,U)}, \varepsilon\rangle) = \langle\text{SBUW}, \varepsilon\rangle$
   $\text{BD}^{-1}(\langle \Diamond_{(B,S)}\Box_{(W,U)}, \varepsilon\rangle) = \langle\sim\text{SB}\sim\text{UW}, \varepsilon\rangle$
   $\Longrightarrow$ inconsistent(SB∼UW) = $\{\langle\text{SBUW}, \varepsilon\rangle\}$

2. BD(SBUB) = $\langle \Box_{(B,S)}\Box_{(B,U)}, \varepsilon\rangle$

   (a) INC($\Box_{(B,S)}\Box_{(B,U)}, \Box_{(B,S)}\Diamond_{(B,U)}$)
   INC($\Box_{(B,S)}\Box_{(B,U)}, \Diamond_{(B,S)}\Box_{(B,U)}$)
   INC($\Box_{(B,S)}\Box_{(B,U)}, \Diamond_{(B,S)}\Diamond_{(B,U)}$)
   $\text{BD}^{-1}(\langle \Box_{(B,S)}\Diamond_{(B,U)}, \neg\rangle) = \langle\text{SB}\sim\text{UB}, \varepsilon\rangle$
   $\text{BD}^{-1}(\langle \Diamond_{(B,S)}\Box_{(B,U)}, \neg\rangle) = \langle\sim\text{SB}\sim\text{UB}, \neg\rangle$
   $\text{BD}^{-1}(\langle \Diamond_{(B,S)}\Diamond_{(B,U)}, \neg\rangle) = \langle\sim\text{SBUB}, \varepsilon\rangle$
   $\Longrightarrow$ inconsistent(SBUB) = $\{\langle\text{SB}\sim\text{UB}, \varepsilon\rangle\}$

   (b) FOR($\Box_{(B,S)}\Box_{(B,U)}, \Box_{(B,S)}\Diamond_{(B,U)}$)
   FOR($\Box_{(B,S)}\Box_{(B,U)}, \Diamond_{(B,S)}\Box_{(B,U)}$)
   FOR($\Box_{(B,S)}\Box_{(B,U)}, \Diamond_{(B,S)}\Diamond_{(B,U)}$)
   $\text{BD}^{-1}(\langle \Box_{(B,S)}\Diamond_{(B,U)}, \varepsilon\rangle) = \langle\text{SB}\sim\text{UB}, \neg\rangle$
   $\text{BD}^{-1}(\langle \Diamond_{(B,S)}\Box_{(B,U)}, \varepsilon\rangle) = \langle\sim\text{SB}\sim\text{UB}, \varepsilon\rangle$
   $\text{BD}^{-1}(\langle \Diamond_{(B,S)}\Diamond_{(B,U)}, \varepsilon\rangle) = \langle\sim\text{SBUB}, \neg\rangle$
   $\Longrightarrow$ forward(SBUB) = $\{\langle\text{SB}\sim\text{UB}, \neg\rangle\}$

From the definitions of BD, BD^{-1}, FOR, DER, and INC, the following corollary is immediately obtained:

Corollary 7

- *For all* $\langle AT', sig \rangle \in$ *forward*(AT) *and all content expressions ac,* $\mathcal{M}(AT)\, ac$ *entails* $\mathcal{M}(AT')\, sig\, ac$.
- *For all* $\langle AT', sig \rangle \in$ *derivable*(AT) *and all content expressions ac,* $\mathcal{M}(AT)\, ac$ *is derivable from* $\mathcal{M}(AT')\, sig\, ac$.
- *For all* $\langle AT', sig \rangle \in$ *inconsistent*(AT) *and all content expressions ac,* $\mathcal{M}(AT)\, ac$ *is inconsistent with* $\mathcal{M}(AT')\, sig\, ac$.

In terms of type-internal expressions, these statements can be reformulated as

- *For all* $\langle AT', sig \rangle \in$ *forward*(AT) *and all content expressions ac,* $AT{:}ac$ *entails* $AT'{:}sig\, ac$.
- *For all* $\langle AT', sig \rangle \in$ *derivable*(AT) *and all content expressions ac,* $AT{:}ac$ *is derivable from* $AT'{:}sig\, ac$.
- *For all* $\langle AT', sig \rangle \in$ *inconsistent*(AT) *and all content expressions ac,* $AT{:}ac$ *is inconsistent with* $AT'{:}sig\, ac$.

Example 5.6:

1. derivable(SB∼UW) $= \{\langle \mathrm{SBUW}, \neg \rangle\}$; hence, SB∼UW:*ac* is derivable from SBUW:¬*ac*.
2. inconsistent(SB∼UW) $= \{\langle \mathrm{SBUW}, \varepsilon \rangle\}$; hence, SB∼UW:*ac* is inconsistent to SBUW:*ac*.
3. forward(SBUB) $= \{\langle \mathrm{SB{\sim}UB}, \neg \rangle\}$; hence, SBUB:*ac* entails SB∼UB:¬*ac*.
4. inconsistent(SBUB) $= \{\langle \mathrm{SB{\sim}UB}, \varepsilon \rangle\}$; hence, SBUB:*ac* is inconsistent with SB∼UB:*ac*. In fact, it is also inconsistent with SBUB:¬*ac*, but this case can be handled by standard type-internal reasoning. Only the first case is specific to reasoning with negative assumption types.

Based on the sets derivable(AT), forward(AT), and inconsistent(AT), special functions for reasoning with negative assumption types can be defined. They permit derivability checks, forward derivation, and consistency checks within the limits that are set by negative assumption types. The functions *na-derivable*, *na-forward*, and *na-consistent* are defined in Table 5.1. All these functions invoke

- *na-derivable*(UMKB, AT:ac):

  loop for $\langle AT', sig\rangle \in$ derivable(AT)
    if **derivable**$_{TI}$(UMKB, AT':$sig\, ac$) then return(true)
  return(false)

- *na-forward*(UMKB, AT:ac):

  loop for $\langle AT', sig\rangle \in$ forward(AT)
    forward$_{TI}$(UMKB, AT':$sig\, ac$)

- *na-consistent*(UMKB, AT:ac):

  loop for $\langle AT', sig\rangle \in$ inconsistent(AT)
    if **derivable**$_{TI}$(UMKB, AT':$sig\, ac$) then return(false)
  return(true)

Table 5.1: Reasoning with negative assumption types

access functions of the TI level on type-internal expressions AT':$sig\, ac$, with the assumption type AT' and the sign sig of the assumption content depending on the $\langle AT', sig\rangle$ elements of derivable(AT), forward(AT), and inconsistent(AT), respectively. In the specifications of Table 5.1, the assumption content ac is again identified with its FOPC correspondence $\tau(ac)$. So, $sig\, ac$ seems to make sense no matter whether $sig = \neg$ or $sig = \varepsilon$. However, it is possible that in content formalisms F without negation operator, $\neg ac = \neg\tau(ac)$ cannot be retranslated into an expression of $\mathcal{L}_F$. So, for calls to TI access functions with argument AT:$\neg ac$ it is possible that $F(\neg ac)$ cannot be determined or is different from $F(ac)$.

It is important to note that the sets derivable(AT), forward(AT), and inconsistent(AT) are finite. That is, they can be exhaustively computed at the time when the assumption type AT is defined to be part of an AsTRa knowledge base. In this case, in the functions defined in Table 5.1 no effort is required to compute the sets derivable(AT), forward(AT), and inconsistent(AT), if this has been done a priori. Therefore, an AsTRa implementation is recommended to compute the sets derivable(AT), forward(AT), and inconsistent(AT) once when the assumption type AT is defined, and to maintain them along with the assumption type in the UMKB.

The special reasoning functions for negative assumptions handle type-internal expressions AT:ac, with AT being a positive or negative assumption type. Therefore, these functions can be used to establish a new type-internal reasoning level for AsTRa systems with negative assumption types. This reasoning level is called

NA, since it handles negative assumptions in a specific way. It is important to note that the reasoning capabilities of this level are based on assuming the D axiom to hold in the modal logic $AL^{\neg}$.

Definition 5.27 (negative assumption reasoning level NA) *NA is a type-internal reasoning level that permits reasoning with negative assumptions. Its reasoning functions are defined as follows:*

- $\mathbf{derivable}_{NA}(UMKB, AT{:}ac) :=$ *na-derivable*$(UMKB, AT{:}ac)$
- $\mathbf{forward}_{NA}(UMKB, AT{:}ac) :=$ *na-forward*$(UMKB, AT{:}ac)$
- $\mathbf{consistent}_{NA}(UMKB, AT{:}ac) :=$ *na-consistent*$(UMKB, AT{:}ac)$

At level NA, only those mechanisms are employed that have been defined based on the special semantics of negative assumption types. Particularly in the case of positive assumption types, the basic type-internal reasoning mechanisms of level TI could be applied in addition to the NA functions. Thus, a further reasoning level is possible in an AsTRa system with negative assumption types. It combines basic type-internal reasoning and specialized negative assumption type reasoning and hence is called $TI + NA$. The reasoning functions of this level will first apply the purely type-internal functions of level TI and only invoke negative type reasoning if necessary.

Definition 5.28 (negative assumption reasoning level $TI + NA$)
$TI + NA$ is a type-internal reasoning level that permits reasoning with negative assumptions. Its reasoning functions are defined as follows:

- $\mathbf{derivable}_{TI+NA}(UMKB, AT{:}ac) :=$
  $\mathbf{derivable}_{TI}(UMKB, AT{:}ac)$ *or* $\mathbf{derivable}_{NA}(UMKB, AT{:}ac)$
- $\mathbf{forward}_{TI+NA}(UMKB, AT{:}ac) :=$
  $\mathbf{forward}_{TI}(UMKB, AT{:}ac)$ *and* $\mathbf{forward}_{NA}(UMKB, AT{:}ac)$
- $\mathbf{consistent}_{TI+NA}(UMKB, AT{:}ac) :=$
  $\mathbf{consistent}_{TI}(UMKB, AT{:}ac)$ *and* $\mathbf{consistent}_{NA}(UMKB, AT{:}ac)$

5.4.4 Integration into Modal Logic Reasoning

Negative assumption types have been introduced as an extension to the basic AsTRa framework, establishing an alternative to the representation of negative assumptions as formulas of the modal logic AL^+. They are particularly useful, if negative assumptions are the only kind of knowledge that goes beyond the possibilities of basic AsTRa. Nevertheless, coexistence of negative assumption types and AL^+ formulas for other type-external knowledge is possible in a UMKB.

It has been shown that contents of all kinds of assumption types can be expressed as modal formulas, namely as formulas of the AL^+ subset $AL^\neg$. Hence, the AL^+ reformulation of the UMKB, UMKB^{AL^+}, is the union of all $AL^\neg$ forms of positive and negative assumptions and all AL^+ formulas that are explicitly present in the current UMKB. The modal logic reasoning option of extended AsTRa, which is based on the AL^+-*derivable*, AL^+-*forward* and AL^+-*consistent* functions, is not influenced by the presence of negative assumption types and remains unchanged.

5.5 Summary

The AsTRa framework for representation and reasoning in user modeling shell systems combines the expressive power and formal semantics of modal logic with the flexibility and practical advantages of a partition-based representation. There are two main parts of the framework:

1. The basic framework allows for the representation of type-internal expressions, i.e., of assumption contents within separate assumption type knowledge bases. Due to the generic AsTRa specifications, a wide range of assumption types, of corresponding stereotyped assumptions, and of content formalisms are possible. UMKB access and reasoning is available via access and reasoning functions of content formalisms. We have characterized the basic framework in terms of modal logic and so have equipped the approach with clear semantics.

2. The correspondence between basic mechanisms and modal logic is the foundation for the extended framework, which provides general-purpose representation and reasoning with full modal logic, even including modal axioms, as well as specialized facilities for handling negative assumptions. Extended reasoning integrates basic UMKB contents, so that the AsTRa framework unites the partition approach and the modal logic approach to (logic-based) user modeling.

UMKB access and reasoning is available at five different reasoning levels, one of them actually being a "no reasoning" level. Figure 5.3 shows the relationships between the five levels. They are coarsely placed into a two-dimensional space, the dimensions being expressive power and inference power. The levels 0 and TI belong to the basic AsTRa framework, while NA, $TI + NA$, and AL^+ are extended reasoning levels. An arrow from one level to a second one indicates that the second one subsumes the first one according to the range of expressions it can process and to the deductions it is capable of. Note that levels NA and TI cannot be related: The reasoning functions of level NA consider the contents of more than one assumption type, but TI can draw inferences within an assumption type which NA cannot draw.

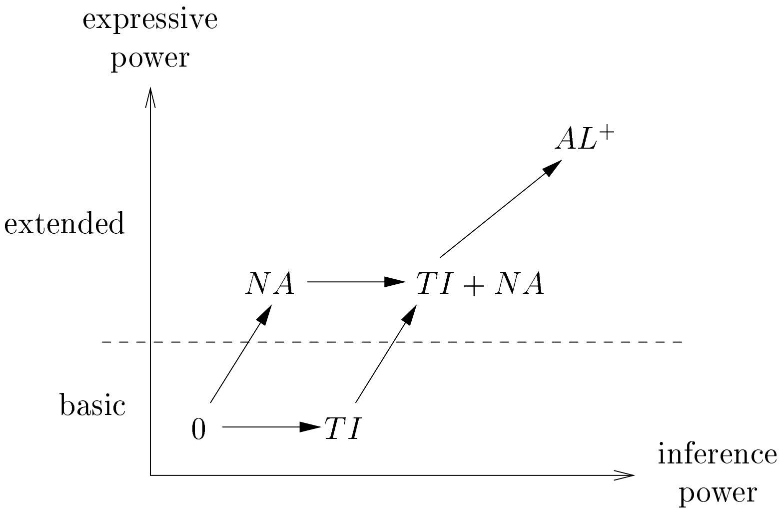


Figure 5.3: An overview of all AsTRa reasoning levels

In an AsTRa system, a wide variety of UMKB contents can be represented; in particular, all logic-based formalisms that were discussed in Chapter 3 can be captured by an AsTRa system. As to inferences, both backward-directed and forward-directed inferences are possible on each reasoning level. Type-internal inferences process assumptions about the user, except if type SB is concerned, and hence can be used for simulative inferences. Meta-level inferences can be performed using the extended mechanisms: type-external expressions typically represent meta-level knowledge of the system. Since AsTRa reasoning takes place within the UMKB, it is mainly suitable for secondary inferences. The only way to do primary inferences would be to code observations about the user within the UMKB, e.g., within the SB type or a special assumption type, and to formulate acquisition rules as (type-external) formulas.

Chapter 6

Powerful and Flexible User Model Representation in BGP-MS

6.1 Introduction

This chapter describes the logic-based representation and reasoning mechanisms that have been developed for the user modeling shell system BGP-MS. As a general and application-independent tool, a shell system is demanded to satisfy the representation needs of a variety of user modeling systems. Therefore, it is important that the representation facilities of a shell provide powerful general-purpose tools as well as the possibility to flexibly choose from the inherent mechanisms more specialized tools that may have less expressive and inferential power but are suitable and sufficient for the needs of a specific system. Like all logic-based representation systems, a logic-based user modeling representation system needs to provide tell and ask functions with reasoning mechanisms like consistency checking or backward-directed reasoning. Moreover, forward-directed reasoning can support user model construction. A user modeling shell must implement these functionalities while considering specific user modeling demands. In its realization of both user model representation and reasoning, it should also consider standard user modeling techniques like the use of stereotypes.

In order to satisfy these demands, the representation and reasoning component of BGP-MS implements the AsTRa framework. The partial knowledge bases of assumption types are naturally realized with a partition mechanism, which is a descendant of the mechanism that was already used in early versions of BGP-MS [Kobsa, 1990]. Hence, the assumption type representation of BGP-MS will go beyond the AsTRa specifications (without violating them) by incorporating the partition inferences inheritance and propagation. As it was demonstrated in [Kobsa, 1990], partition inheritance hierarchies are beneficial for several user modeling purposes, among them representation of shared or mutual beliefs, but also representation and maintenance of stereotypes. In Section 6.2, it will be shown how partition hierarchies were utilized to implement assumption types. Based on the modal logic semantics of the AsTRa framework, partition inheritance and propagation can be characterized in a formal way.

In an AsTRa system, the range of possible assumption types is clearly defined and determined by the set of available modalities. BGP-MS starts with the standard modalities B and W for beliefs and goals, but the developer of a user modeling system can add further modalities to this set. Stereotyped assumptions may also concern different modalities. The BGP-MS stereotype representation was extended according to the AsTRa definitions, in order to allow more than just belief assumptions in stereotypes.

In contrast to earlier versions of BGP-MS, partition maintenance was made independent of any formalism for the representation of partition contents. Thus, the integration of several content formalisms became possible. SB-ONE, a KL-ONE-like language for terminological, concept-based representation, has always been a part of BGP-MS [Kobsa, 1990]. In Section 3.3.2, several examples of user modeling systems were described that make use of a concept-based representation. Thus proven useful, SB-ONE was retained as one assumption content formalism of BGP-MS. However, concept-based languages like SB-ONE are limited concerning their expressive power. Disjunction, negation, quantification and implication can only be used in restricted forms. First-order predicate calculus offers all these facilities. Therefore, FOPC was chosen as second content formalism [Kobsa and Pohl, 1995]. FOPC reasoning was implemented based on the automated theorem prover OTTER [McCune, 1994], which was extended with an interface for communicating with other parts of BGP-MS. In terms of the AsTRa framework, SB-ONE is a simple formalism (cf. Section 5.2.4) with tell and ask functions only, while OTTER was used to implement FOPC as a full AsTRa content formalism. Section 6.3 describes the two content formalisms.

BGP-MS also implements a modal logic extension to assumption types, following the template of the extended AsTRa framework. Modal (logic) reasoning is realized with techniques for translating modal predicate logic into first-order predicate calculus, so that OTTER also becomes applicable for modal reasoning. The translation methods allow to process arbitrarily complex formulas of the extended assumption type logic AL^+. Hence, also formula schemes, i.e. modal logic axioms, can be employed to enhance the reasoning capabilities of BGP-MS even further. A crucial problem is the integration of type-internal AsTRa expressions, which are represented as partition contents, into the modal reasoning process. Since type-internal expressions correspond to modal formulas, these corresponding formulas can be translated into FOPC like all other modal formulas in the UMKB. Specific properties of the translation technique are exploited to perform this translation in an efficient manner, while taking into account as well as taking advantage of the built-in propagation and inheritance inferences of the partition mechanism. Section 6.4 describes the implementation of modal reasoning and its integration with the partition mechanism in detail.

Negative assumption types were introduced in the extended AsTRa framework as a specialized means for handling negative assumptions. In BGP-MS, also negative assumption types are implemented with partitions. The AsTRa specification is strictly obeyed. Section 6.5 deals with the implementation of negative assumption types in BGP-MS and describes their integration into the modal reasoning mechanisms of BGP-MS.

On the one hand, the modal logic extension equips BGP-MS with very powerful representation and reasoning capabilities. On the other hand, BGP-MS inherits flexibility and scalability from the AsTRa framework. A BGP-MS user, i.e. the developer of a user modeling system, has a considerable variety of choices concerning user model representation and reasoning. The range of possible assumption types can be extended by adding modalities, assumption types can be arranged into inheritance and propagation hierarchies, stereotypes may contain assumptions of different types, negative assumptions may be represented with either negative assumption types or modal logic, modal axioms can be defined to determine modal reasoning behavior, and it is possible to freely switch between type-external (i.e., modal logic) and type-internal reasoning on all AsTRa reasoning levels. In principle, the AsTRa approach of integrating modal logic and assumption types is bottom-up (cf. Section 2.4). But also top-down handling is possible; the user model developer can choose to use modal logic as interface language for UMKB contents and let BGP-MS decide on how a modal expression can be handled appropriately. Section 6.6 further describes the flexibility that is offered by BGP-MS.

6.2 Representing Assumption Types with Partitions

6.2.1 The Partition Mechanism KN-PART

For BGP-MS, the partition mechanism KN-PART [Scherer, 1990; Fink and Herrmann, 1993] was chosen as the basis of its AsTRa implementation. The partitions of KN-PART are very close to assumption types, since they basically provide the possibility to store a set of content items. Partition contents normally are expressions of a knowledge representation formalism, so that a partition forms a partial knowledge base of a whole knowledge-based system. However, in principle partition contents can be arbitrary data. KN-PART internally employs a database mechanism that offers functions for entry and retrieval. These facilities must be used by content formalisms for the implementation of $store_F$ and $fetch_F$ functions.

In contrast to other partition-based systems like those described in [Cohen, 1978] and [Ballim and Wilks, 1991b], KN-PART does not know nested partitions for the explicit representation of belief nestings. In general, KN-PART can be seen independent of the task of belief or user model representation. It provides the means that were used in BGP-MS for implementing the AsTRa framework, and it has even more. With KN-PART, it is not only possible to establish a set of isolated partitions, each of which represents an assumption type. KN-PART was enhanced by a subordination relation that can be introduced between partitions. Thus, the definition of partition hierarchies becomes possible, which offer built-in inferences to the developer of a user modeling system. These inferences are *inheritance* and *propagation*. Common partition contents of all subordinate partitions of a partition P will be propagated to P, after a new entry into one of the subordinate partitions was made. The other way round, superordinate partitions inherit their contents to all subordinates at the time of retrieval.

In the following, we will identify a partition P with its set of partition contents, the *partition knowledge base* KB_P. A partition hierarchy is defined as follows:

Definition 6.1 (partition hierarchy) *A partition hierarchy is a pair $\langle \mathcal{P}, \mathcal{H} \rangle$, where*

1. $\mathcal{P} = \mathcal{P}_{St} \uplus \mathcal{P}_{Pr}$ *is a set of partitions. A partition $P \in \mathcal{P}_{St}$ is called* standard partition*, while a partition $P \in \mathcal{P}_{Pr}$ is called* primitive partition.

2. $\mathcal{H}$ *is a set of partition hierarchy links (P_1, P_2). P_1 is a subordinate partition (short:* subpartition*) of P_2, P_2 is a superordinate partition (short:* superpartition*) of P_1.*

A partition hierarchy is a directed acyclic graph.

Standard and primitive partitions differ with respect to propagation. Only a standard partition gets common contents of its subpartitions propagated. If for the subpartitions $P_1, \ldots, P_n$ of a standard partition P_s, all partition knowledge bases $\mathrm{KB}_{P_1}, \ldots, \mathrm{KB}_{P_n}$ have a common content *pc*, i.e. $pc \in \mathrm{KB}_{P_i}$ for $i = 1, \ldots, n$, then *pc* is propagated to P_s. That is, *pc* is removed from all KB_{P_i} and entered into KB_{P_s}. Propagation is performed after an entry into a partition knowledge base was made. Hence, if a partition content can be propagated, the process will repeat recursively.

Inheritance occurs, when a content is to be retrieved from a partition. Then, a partition knowledge base KB_P inherits the contents of all direct and indirect superpartitions, i.e. of all partition knowledge bases P' with (P, P') being an element of $\mathcal{H}^*$, the transitive closure of the hierarchy link set $\mathcal{H}$. As far as inheritance is concerned, other terms are often used instead of 'partition'. Leaf partitions in a hierarchy (with all their directly or indirectly inherited contents) are called *views*, while a non-leaf partition is called *vista* [Hendrix, 1979]. We will use 'view' in both cases, unless the distinction is relevant. A view can also be seen as a set V of partitions that contains its base partition P_V and all direct and indirect superpartitions. Then, a *view knowledge base* KB_V is the union of the knowledge bases of all partitions of the view:

$$\mathrm{KB}_V := \bigcup_{P \in V} \mathrm{KB}_P$$

Entering a new content into a view knowledge base KB_V means entering it into the knowledge base of the base partition KB_{P_V}. *Propagation* does not remove contents from a view knowledge base, but changes the location of the content within the partition set V. Let ac be a content of a view knowledge base KB_V, stored in partition $P \in V$. When ac is propagated to a superpartition P' of P, it is still contained in KB_V since also $P' \in V$, but it has been added to the view V' the base partition of which is P' (i.e., $P_{V'} = P'$). If views are regarded as partition sets, *inheritance* means that view knowledge bases are dynamically composed from the concerned partition knowledge bases. We will see that this dynamic composition will be important for the purposes of BGP-MS.

6.2.2 Assumption Types

Given the possibilities of KN-PART, assumption types are represented with the help of partitions and views. Each type AT will be identified with its own partition $P(AT)$, which may be linked hierarchically to other partitions. Hence, inheritance and propagation relationships can be established between assumption types. Typically, an assumption type that expresses mutual beliefs will be defined as superordinate type that inherits to others and gets their common contents propagated. E.g., a developer may introduce the type SBMB as inheriting to SB and SBUB. Such a definition leads to the simple partition hierarchy depicted on the left of Figure 6.1. For each assumption type, a partition is created, and the SBMB partition becomes superordinate to the SB and SBUB partitions.

Since an assumption type AT is identified with a single partition $P(AT)$, there is also a corresponding view $V(AT)$ for each type with $P_{V(AT)} = P(AT)$. The assumption type knowledge base, KB_{AT}, is implemented by the view knowledge base $\mathrm{KB}_{V(AT)}$. For content formalisms, this means that their knowledge base access functions need to operate on the view knowledge base $\mathrm{KB}_{V(AT)}$ in order to access an assumption type knowledge base KB_{AT}. This is demonstrated for $store_F$ and $derivable_F$; other access functions work analogously.

- $store_F(\mathrm{KB}_{AT}, ac) := store_F(\mathrm{KB}_{V(AT)}, ac)$
- $derivable_F(\mathrm{KB}_{AT}, ac) := derivable_F(\mathrm{KB}_{V(AT)}, ac)$

So, in BGP-MS assumption types can be identified with their corresponding views. In the following, both notions will be used interchangeably. So, we will also use $V{:}ac$ instead of $AT{:}ac$ for type-internal (i.e., view-internal) expressions, particularly if it shall become clear that we deal with issues specific to BGP-MS.

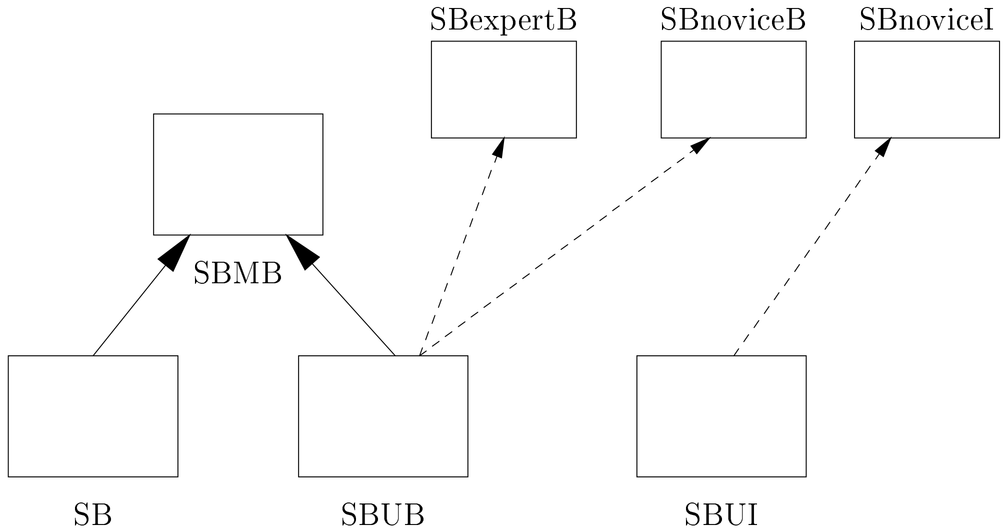


Figure 6.1: Assumption types and stereotypes in a partition hierarchy

6.2.3 Stereotypes

The AsTRa framework defines a stereotype S as a set of pairs

$$S = \{\langle \mathrm{KB}_1, AT_1 \rangle, \ldots, \langle \mathrm{KB}_n, AT_n \rangle\}$$

where KB_i is a (stereotype) knowledge base and AT_i is its associated assumption type. With the partition mechanism KN-PART at hand, the implementation of this definition is straightforward. In BGP-MS, a stereotype S is represented by a set of primitive partitions $P_1^S, \ldots, P_n^S$. Each stereotype!partition is associated with an assumption type. Stereotype contents are distributed across the partition knowledge bases of the stereotype partitions, depending on which assumption type they belong to. Typically, a stereotype contains direct assumptions about user beliefs, goals, preferences, etc. Therefore, stereotype partitions are typically associated with assumption types like SBUB, SBUW, etc.

If a stereotype applies to the current user, the contents of stereotype partitions must be integrated into the appropriate assumption type knowledge bases. In this case, each stereotype partition P_i^S will become superordinate partition of the corresponding assumption type partition $P(AT_i)$. Thus, the assumption type view is extended by the stereotype partition: $V(AT_i) := V(AT_i) \cup \{P_i^S\}$. Through the dynamic composition of view knowledge bases, the assumption type knowledge base will automatically inherit the contents of the stereotype partition at the time of subsequent retrieval operations. Note that stereotype partitions are primitive partitions. This prevents assumption contents from getting propagated into stereotype partitions. This is reasonable because when a stereotype is no longer applicable to a user, the hierarchy links to its stereotype partitions will be removed. Then, propagated assumption contents would no longer be contained in the view knowledge bases of the associated assumption types.

In the current implementation of BGP-MS, stereotype partitions can be linked to all assumption types. Typically, stereotype partitions will be linked to assumption types that are related to the user. The reason for this is quite obvious: stereotypes contain assumptions about user groups, which can be ascribed to individual users; pure system beliefs do not depend on the current classification of the user. Furthermore, stereotype partitions may be linked to partitions concerning mutual beliefs. According to [Clark and Marshall, 1981], one type of mutual knowledge is based on the "community membership" of two dialog partners; knowledge that is universal within one particular group or community can be assumed to be mutually believed if two dialog partners mutually know they belong to this community. This principle can be applied in BGP-MS by predefining possible mutual beliefs in stereotypes, albeit it may appear problematic to say that system and user belong to one community and that they mutually know this. But imagine, for instance, a tutoring system: both system and a very advanced user might be regarded as belonging to the community of experts in the tutoring domain or one of its subdomains. However, in previous applications of BGP-MS, stereotyped mutual belief was not made use of.

In BGP-MS, a stereotype is introduced by defining a new stereotype name and listing the set of assumption types that the stereotype partitions are to be associated with. Then, stereotype partitions will be created automatically; their partition label is generated from the corresponding assumption type label by replacing each occurrence of U with the stereotype name. These names can then be used to address the stereotype partitions, mainly for filling them with stereotyped assumptions. For instance (let the modality I stand for "interests"), if a stereotype 'novice' is defined to be related to assumption types SBUB and SBUI, two stereotype partitions are created and labeled SBnoviceB and SBnoviceI. Thus, an AsTRa stereotype

$$\text{novice} = \{\langle \text{KB}_{\text{SBnoviceB}}, \text{SBUB}\rangle, \langle \text{KB}_{\text{SBnoviceI}}, \text{SBUI}\rangle\}$$

is introduced into the current UMKB.

Illustrating this example, Figure 6.1 shows partitions of two stereotypes that collect the presumable beliefs of experts and the presumable beliefs and interests of novices. The partitions are associated with assumption types SBUB and SBUI, respectively. So, they are labeled SBexpertB, SBnoviceB, and SBnoviceI. They will get linked to the assumption type base partitions SBUB and SBUI, respectively, if their stereotype is activated (this is symbolized by dashed lines).

6.2.4 Partition Hierarchies and Modal Logic

In section 5.2.6, the relationship between the basic AsTRa framework and the restricted modal logic AL was described. The range of BGP-MS assumption types strictly obeys the AsTRa specification. So, syntactically this relationship also exists in the BGP-MS implementation of the framework, presumed that BGP-MS content formalisms are logic-based.

Semantically, the situation is not that simple: BGP-MS exceeds the AsTRa specifications due to its partition mechanism, which allows the representation of propagation and inheritance relationships between assumption types. These relationships and the corresponding inferences are not automatically covered within AL. Propagation and inheritance are general inference mechanisms that are independent of the contents of partitions. In modal logic, general inference rules are usually established with the help of axioms. So, it is near at hand to define axioms that cover propagation and inheritance. For each assumption type AT with subordinate types $AT_1, \ldots, AT_n$ a *hierarchy axiom*

$$\mathcal{M}(AT)\Phi \leftrightarrow \mathcal{M}(AT_1)\Phi \wedge \ldots \wedge \mathcal{M}(AT_n)\Phi$$

can be introduced. The $\rightarrow$ direction is to describe inheritance, and the $\leftarrow$ direction is to describe propagation. Such axioms would be added to the UMKB^{AL} that corresponds to a BGP-MS UMKB in order to achieve equivalence again.

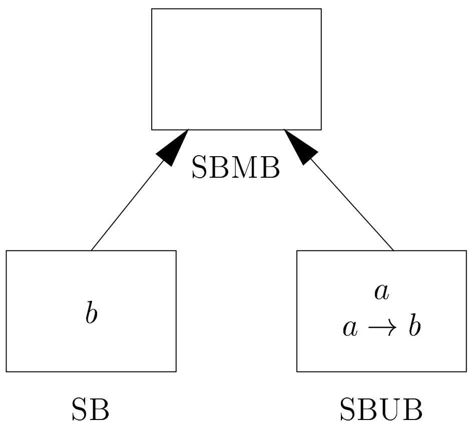


Figure 6.2: Assumption types and stereotypes in a partition hierarchy

However, if extended with hierarchy axioms, assumption logic now allows more inferences than view-internal reasoning in BGP-MS, as it has been described so far. The problem is illustrated by Figure 6.2. It shows a simple partition hierarchy with three partitions. Partition contents are expressions of propositional calculus (i.e., we assume a content formalism PC that handles propositional formulas). This UMKB corresponds to the assumption logic expressions

$$\{\Box_{(B,S)}b, \Box_{(B,S)}\Box_{(B,U)}a, \Box_{(B,S)}\Box_{(B,U)}(a \rightarrow b)\}$$

plus the hierarchy axiom

$$\Box_{(B,S)}\Box_{(B,M)}\Phi \leftrightarrow \Box_{(B,S)}\Phi \wedge \Box_{(B,S)}\Box_{(B,U)}\Phi \tag{6.1}$$

From this set of formulas, we can derive $\Box_{(B,S)}\Box_{(B,M)}b$ as follows: From $\Box_{(B,S)}\Box_{(B,U)}a$ and $\Box_{(B,S)}\Box_{(B,U)}(a \to b)$, $\Box_{(B,S)}\Box_{(B,U)}b$ can be inferred using modal modus ponens (see p.111). Together with $\Box_{(B,S)}b$ and the hierarchy axiom (6.1) this leads to $\Box_{(B,S)}\Box_{(B,M)}b$. However, $\mathbf{ask}_{TI}(\mathrm{UMKB}, \textsf{SBMB:b})$ would return false, since $\mathit{derivable}_{PC}(\mathrm{KB}_{SBMB}, b)$ cannot infer b from the (empty) vista SBMB. The reason is that propagation is weaker than the $\leftarrow$ direction of hierarchy axioms: It does not consider intermediate inference results, but only works when a new expression is "told". Inheritance is less problematic, since it works at query time to compute $\mathrm{KB}_{AT} = \mathrm{KB}_{V(AT)}$, so that inherited view contents are completely available.

There are several possibilities to cope with the propagation problem. First, view-internal forward reasoning could be exhaustively used at tell time to infer implicit view contents, which then would be entered into views and hence be considered by the propagation mechanism. Second, the $\leftarrow$ direction of hierarchy axioms could be restricted to explicit view contents. This would characterize view-internal reasoning as described so far; however, such a formalization cannot be done using the formal apparatus of assumption logic. For BGP-MS, a third alternative was chosen, namely to modify view-internal reasoning such that it better corresponds to hierarchy axioms.

The propagation problem is due to the fact that common contents of a set of subordinate partitions, which are only implicitly contained in one of the partitions, are not propagated. Therefore, if a query refers to a vista (a non-leaf partition), common *derivable* contents of subordinate views are missed. However, for a separate query to each of these views, the result would be positive. As a consequence, the AsTRa **ask** function was redefined for BGP-MS on the type-internal reasoning level TI. Let $\{V_1, \ldots, V_n\}$ be the set of all views (not vistas) that directly or indirectly inherit from (and propagate to) $V(AT)$[1]. A type-internal query concerning AT will be redirected to all these views. I.e.,

$$\begin{aligned}&\mathbf{ask}(TI, \mathrm{UMKB}, AT{:}ac) := \\ &\qquad \bigwedge_{i=1,\ldots,n} \Big(\mathit{fetch}_{F(ac)}(\mathrm{KB}_{V_i}, ac) \text{ or } \mathit{derivable}_{F(ac)}(\mathrm{KB}_{V_i}, ac)\Big)\end{aligned}$$

With this redefinition given, a query for SBMB:b will be successful, when posed to the example UMKB of Figure 6.2:

$$\begin{aligned}\mathbf{ask}(TI, \mathrm{UMKB}, \textsf{SBMB:b}) &= \mathit{fetch}_{PC}(\mathrm{KB}_{SB}, b) \wedge \mathit{derivable}_{PC}(\mathrm{KB}_{SBUB}, b) \\ &= \text{true} \qquad\qquad\quad \wedge \text{ true} \\ &= \text{true}\end{aligned}$$

[1]If $V(AT)$ is a leaf partition, then $V(AT)$ will be the only member of this set.

6.3 Representing Assumption Contents

6.3.1 Representing Conceptual Knowledge with SB-ONE

The conceptual knowledge representation language SB-ONE [Kobsa, 1990; Kobsa, 1991] can be used to formulate assumptions concerning the concepts or terminology of an application domain. SB-ONE loosely fits into the KL-ONE paradigm [Brachman, 1978; Brachman and Schmolze, 1985]. Its main representation elements are (general) concepts and attribute descriptions ("role descriptions" in terms of KL-ONE). A concept describes a class of domain objects, which is defined by its attribute descriptions and its subsumption relations to other concepts. An attribute (a role) defines a relation between objects of the concerned class and objects of the same or a different class. A subsumption relation basically describes a subset relationship between two classes. In SB-ONE, a subsumption link, i.e. an IS-A link, between two concepts implies that the subset concept inherits all attribute descriptions of the superset concept.

So, the central part of any SB-ONE knowledge base is a subsumption hierarchy of general concepts, the root of which is a universal thing concept. Attribute descriptions define further relationships between concepts. An example is the value restriction of attributes, which defines that the second argument of the role relation is restricted to a certain concept, i.e. to objects of a certain class.

SB-ONE implements a wide range of constructs for expressing concept-based knowledge. In BGP-MS, only a subset of these constructs is employed, which comprises standard constructs of modern terminological formalisms, namely description logics. In Section 3.3.1, examples of representation elements of description logics were given. In addition, a prototype interface language for description logic elements was presented there. The interface language that can be used in BGP-MS to specify SB-ONE constructs follows this proposal. Possible expressions of $\mathcal{L}_{SB\text{-}ONE}$ are listed in Table 6.1, together with corresponding expressions of description logic and their translations into FOPC. The FOPC translations induce the translation function $t_{SB\text{-}ONE}$, which makes SB-ONE a (at least pragmatically) logic-based AsTRa content formalism. C, C', C_i, and R are concept and role variables, which are to be replaced by concrete concept and role symbols when forming $\mathcal{L}_{SB\text{-}ONE}$ expressions.

There are several built-in inferences in SB-ONE, mainly based on the inheritance and transitivity property of subsumption links. E.g., assume there is a concept C_1 with an attribute relation R that is value-restricted to a concept C_2. A third concept C_3 that is subsumed by C_1 (i.e., (isa C_3 C_1)) inherits the attribute R together with its value restriction. Furthermore, if another concept C_4 is subsumed by C_3, SB-ONE will infer that it is indirectly also subsumed by C_1. In addition, SB-ONE provides an inference process that can be explicitly invoked, namely the classifier. It can be used to determine the most specific subsuming concept for a given concept. The classifier of SB-ONE is a knowledge engineering

$\mathcal{L}_{SB\text{-}ONE}$	*descr. logic*	*FOPC translation*
(isa C C')	$C \sqsubseteq C'$	$\forall x\, C(x) \rightarrow C'(x)$
(isa C (and $C_1 \ldots C_n$))	$C \sqsubseteq C_1 \sqcap C_n$	$\forall x\, C(x) \rightarrow C_1(x) \wedge \ldots \wedge C_n(x)$
(isa C (all R C'))	$C \sqsubseteq \forall R.C'$	$\forall x, y\, C(x) \wedge R(x,y) \rightarrow C'(y)$
(concept C)	$C \sqsubseteq$ thing	$\forall x\, C(x) \rightarrow \mathsf{thing}(x)$

Table 6.1: $\mathcal{L}_{SB\text{-}ONE}$ expressions and their FOPC translations

tool. In BGP-MS, this procedure can be useful, if the developer of an adaptive system wants to define a concept hierarchy as system domain knowledge, i.e. within assumption type SB, or as stereotyped assumptions. Within the individual user model, the classifier is irrelevant, since the assumed conceptual knowledge of the user or her assumed interests in domain concepts are not necessarily optimally structured in a knowledge engineering sense and therefore should not be manipulated by an automatic engineering tool.

For all of its representational constructs, SB-ONE offers definition and retrieval functions that can be used to enter the construct into a given knowledge base or to query if the construct is (perhaps implicitly) contained in the knowledge base. These functions can be used to implement $tell_{SB\text{-}ONE}$ and $ask_{SB\text{-}ONE}$ functions. The basic principle is that both functions determine the SB-ONE construct that is instantiated by the given assumption content ac and then call the corresponding definition or retrieval function, respectively. The same mechanism can be used when, in case of top-down UMKB access in the extended AsTRa framework, assumption contents are specified in FOPC only. Then, the FOPC equivalence patterns of $\mathcal{L}_{SB\text{-}ONE}$ expressions can be used to determine the concerned SB-ONE construct.

Formally, this can be defined as follows:

Definition 6.2 (SB-ONE: tell and ask) *Let con(ac) be the construct that an assumption content ac instantiates, and let def_c and $retr_c$ be the definition and retrieval functions for an SB-ONE construct c. Then*

- $tell_{SB\text{-}ONE}(KB_V, ac) := def_{con(ac)}(KB_V, ac)$
- $ask_{SB\text{-}ONE}(KB_V, ac) := retr_{con(ac)}(KB_V, ac)$

By providing *tell* and *ask* functions only, SB-ONE becomes a simple AsTRa content formalism (cf. Section 5.2.5).

Similar to all KL-ONE-like formalisms, the expressive power of SB-ONE is limited. In particular, the representation of disjunction, negation, quantification and implication is quite restricted, and predicates with an arity of more than two are not allowed. All these constructs, however, are permitted in first-order predicate calculus. In the following section, the implementation of FOPC as additional content formalism for BGP-MS is described.

6.3.2 FOPC Reasoning with OTTER

Because of the above-mentioned limitations of the expressive power of SB-ONE, first-order predicate calculus was chosen as a second content formalism for BGP-MS. Several logic reasoners were examined as to what subset of FOPC they can process and which of the desired reasoning tasks can be realized [Han, 1995]. Finally, the automated theorem prover OTTER (Organized Techniques for Theorem-Proving and Effective Research) [McCune, 1994] turned out to be most flexible and powerful as well as efficient and portable. This section gives an overview of OTTER and describes how the interface functions needed for integrating FOPC into an AsTRa implementation were realized.

OTTER

OTTER is based on the resolution principle [Robinson, 1965]. Resolution can be regarded as a generalization of modus ponens. It requires that logical expressions are available as *clauses*. I.e., FOPC formulas must be transformed into disjunctions of positive or negative literals (i.e., non-negated or negated atoms). Two clauses C_1 and C_2 can be *resolved*, if one contains a literal L and the other its negation $\neg L$. In case of predicate logic, the atoms L can be made identical by *unification*. The result of the resolution step (also called *resolvent*) is a clause C' that is a disjunction of all literals of the parent clauses without L and $\neg L$. C' logically follows from $\{C_1, C_2\}$. A resolution proof is found if the empty clause is derived, which represents a contradiction. That is, resolution is a refutation procedure that essentially can test if a set of FOPC formulas is contradictory. However, refutation procedures can also be used for other purposes. For example, checking if a formula p is derivable from a set of formulas, F, is equivalent to checking if $F \cup \{\neg p\}$ is contradictory.

In its main procedure, OTTER uses the "given clause algorithm", which is a version of the set-of-support strategy [Wos *et al.*, 1965]. This inference strategy is based on the idea that in most refutation problems, there will be a large consistent subset of formulas, and only some formulas may cause the whole set to be contradictory. So, the set-of-support strategy divides a given set of formulas into a consistent subset T and the "set of support" S. In every resolution step, one of the parent clauses must be taken from S, and the resolvents go into S again.

Basically, OTTER gets two sets of clauses as input, called *usable* (the consistent subset) and *sos* (short for set of support). It picks one clause (the "given clause", hence the name of the algorithm) from *sos* and resolves it with as much clauses from *usable* as possible. Afterwards, the given clause moves to *usable*, and the resolvents are added to *sos*. This process is repeated until the empty clause is found or *sos* is empty. In the second case, OTTER has not been able to find a refutation for the given set of clauses. In addition to clauses, OTTER will also accept standard FOPC formulas as input and transform them into clauses.

The refutation procedure of OTTER is flexible enough to be useful for a number of reasoning purposes. As already mentioned above, backward reasoning for answering queries to a knowledge base can be done by taking the knowledge base as *usable* set and the negation of the query as *sos*. If OTTER finds a contradiction, the queried expression is derivable from the knowledge base, and the query can be answered positively. Consistency of a new knowledge base entry with the current knowledge base can be checked by taking the new entry as *sos*, with the knowledge base being the *usable* set again. If OTTER finishes with a proof (the empty clause), the new entry is inconsistent with the knowledge base. With the same constellation, also forward reasoning can be accomplished, since all OTTER resolvents logically follow from the conjunction of the knowledge base and the new entry.

Table 6.2 presents these possibilities using the classical example of men and mortality. First, a small FOPC knowledge base with three formulas and the corresponding *usable* clause set is presented[2]. Then, the use of OTTER for derivability and consistency checks as well as for forward reasoning is illustrated. In all examples, a query or input is given, which is appropriately transformed into an *sos* clause set. The resolvents are the clauses that OTTER generates by applying its algorithm. All clauses are numbered; for resolvents, the numbers of their parent clauses are additionally given in square brackets.

Note that every single OTTER run can be controlled by modifying the values of an extensive list of flags and parameters. For instance, the resolution strategy can be chosen, and time and memory limits can be specified.

KN-OTTER

OTTER is a C program that is designed for batch-style processing of input files and generating output files. Therefore, the program had to be extended and modified in several ways in order to become usable for on-line processing of logical expressions in BGP-MS. Mainly, the interface of OTTER was changed from file input/output processing to inter-process message communication based on the inter-process communication management system KN-IPCMS. Messages to KN-OTTER, as the modified system is called within BGP-MS, can communicate the same data to OTTER as the program would read from input files otherwise. So, they may contain OTTER flag and parameter settings, clause or formula lists, or special KN-OTTER commands. These commands can cause KN-OTTER to

[2]The OTTER notation for clauses is used: Clauses are terminated by `.`, and literals are separated by `|`. Upper case argument symbols are variables.

Sample Knowledge Base:

formulas	*usable clause set*
$\forall x\ \mathrm{man}(x) \rightarrow \mathrm{mortal}(x)$	(1) `-man(X) | mortal(X).`
man(socrates)	(2) `man(socrates).`
¬mortal(superman)	(3) `-mortal(superman).`

1. **Derivability check**

Query:	mortal(socrates)?
sos :	(4) `-mortal(socrates).`
Resolvents:	(5)[4,1] `-man(socrates).`
	(6)[5,2] `.` (empty clause)
Result:	Query is derivable from the knowledge base.

2. **Consistency check**

Input:	man(superman)
sos :	(4) `man(superman).`
Resolvents:	(5)[4,1] `mortal(superman).`
	(6)[5,3] `.` (empty clause)
Result:	Input is inconsistent with the knowledge base.

3. **Forward reasoning**

Input:	man(captainKirk)
sos :	(4) `man(captainKirk).`
Resolvents:	(5)[4,1] `mortal(captainKirk).`
Result:	mortal(captainKirk) follows from man(captainKirk) and the knowledge base.

Table 6.2: Reasoning with OTTER

return diverse results as well as traces of the OTTER proof process (see Fig. 6.3). The main return value is an exit code that states the reason why KN-OTTER finished, namely because of a refutation, of an empty set of support, or of other reasons.

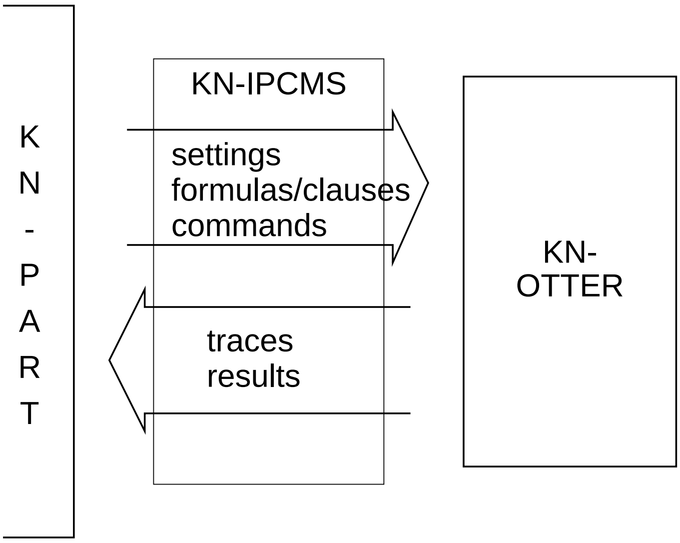


Figure 6.3: Using KN-OTTER for FOPC inferences

BGP-MS makes only little use of the many resolution strategies and algorithms that are available in OTTER. Normally, binary resolution together with factorization[3] is employed; this combination constitutes a theoretically sound and complete inference procedure. Of course, the undecidability of FOPC remains a problem. In BGP-MS, it is dealt with by limiting the run time of OTTER, so that KN-OTTER will always stop. This is achieved by using OTTER control parameters. Normally, one standard setting of OTTER parameters is used.

In order to make KN-OTTER available to BGP-MS, a boolean-valued function *kn-otter* was implemented as interface to KN-OTTER. It receives two parameters, 'usable' and 'sos'[4]. As the parameter names indicate, the 'usable' parameter receives the formulas that are to become the *usable* clause set (typically the current contents of a knowledge base), and the 'sos' parameter receives a query or input expression that is transformed into the *sos* clause set. *kn-otter* returns

[3]Factorization means the unification and merging of literals within one clause. In case of full resolution, this is a part of the resolution step. In case of binary resolution, factorization is not included and therefore has to be done separately.

[4]Actually, users of *kn-otter* can influence all OTTER settings via additional parameters. In BGP-MS, there are standard values for OTTER settings, which are applied in all calls to *kn-otter*. Some of these values differ from default values of OTTER; e.g., a time limit ensures that *kn-otter* returns even if OTTER enters an infinite loop.

true if and only if OTTER finds a refutation within the clause set that results from the 'usable' and 'sos' parameters.

Making FOPC an AsTRa Content Formalism using KN-OTTER

Based on the capabilities of KN-OTTER, FOPC could become a formalism for assumption contents within the AsTRa implementation of BGP-MS. FOPC is a full content formalism, i.e. it provides functions $store_{FOPC}$, $forward_{FOPC}$, $consistent_{FOPC}$, $fetch_{FOPC}$, and $derivable_{FOPC}$. FOPC formulas (not clauses!) are maintained as partition contents by the partition mechanism. So, the basic functions $store_{FOPC}$ and $fetch_{FOPC}$ can be realized without KN-OTTER:

$fetch_{\boldsymbol{FOPC}}(\mathrm{K}B_V, f)$ returns true iff KB_V contains f or a formula f' that differs from f only in its variable symbols.

$store_{\boldsymbol{FOPC}}(\mathrm{K}B_V, f)$ enters formula f into partition P_V, if f is not yet contained in KB_V (i.e., if $\mathrm{fetch}_{FOPC}(\mathrm{KB}_V, f)$ would return false).

In addition, the *kn-otter* function immediately yields an implementation of $derivable_{FOPC}$ and $consistent_{FOPC}$ functions.

$derivable_{\boldsymbol{FOPC}}(\mathrm{K}B_V, f) := \textit{kn-otter}(\mathrm{KB}_V, \{\neg f\})$

$consistent_{\boldsymbol{FOPC}}(\mathrm{K}B_V, f) := \mathrm{not}(\ \textit{kn-otter}(\mathrm{KB}_V, \{f\})\ )$

As illustrated in Table 6.2, forward reasoning with OTTER can be done in a way that is very similar to consistency checking. If *kn-otter* is extended to return not only a boolean value, but also the resolvents it generated during its reasoning process, $forward_{FOPC}$ can make the same call to *kn-otter* as $consistent_{FOPC}$. Let this extended version of *kn-otter* be called *kn-otter**, then we get:

$forward_{\boldsymbol{FOPC}}(\mathrm{K}B_V, f) := \textit{kn-otter*}(\mathrm{KB}_V, \{f\})$

[Schauer and Pohl, 1997] sketches the main ideas of FOPC forward reasoning in BGP-MS, and [Schauer, 1998] describes in detail how *kn-otter** has been implemented. In this implementation, several filters are applied to the set of OTTER resolvents, because this set can be quite large and may contain lots of irrelevant information.

Besides a set of functions, an AsTRa content formalism F needs to define an interface language $\mathcal{L}_F$. As FOPC language $\mathcal{L}_{FOPC}$, the OTTER notation for formulas was chosen. Table 6.3 shows the syntax definition of $\mathcal{L}_{FOPC}$; in comparison to Definition 3.5, & replaces the conjunction symbol $\wedge$, | replaces the disjunction symbol $\vee$, `->` and `<->` are the implication and co-implication arrows, and `-` replaces the negation symbol $\neg$. The quantifiers $\forall$ and $\exists$ are replaced by all and exists. Beyond this definition, formulas are demanded to bind all their variables with quantifiers.

```
<formula> ::= predicate[(<term>+)]
            | <formula> { & <formula> }+
            | <formula> { | <formula> }+
            | <formula> -> <formula>
            | <formula> <-> <formula>
            | –<formula>
            | all variable+ <formula>
            | exists variable+ <formula>
<term>    ::= variable | function[(<term>+)]
```

Table 6.3: Syntax of $\mathcal{L}_{FOPC}$

An example: In $\mathcal{L}_{FOPC}$, the formula[5]

$$\forall x\, \text{shark}(x) \land \neg \text{looks}(x, \text{friendly}) \rightarrow \text{attacks}(\text{flipper}, x)$$

looks like

```
all x shark(x) & –looks(x,friendly) -> attacks(flipper,x)
```

Hybrid Reasoning with KN-OTTER

By definition, an AsTRa content formalism must be logic-based and, hence, correspond to a subset of FOPC. Therefore, full first-order logic is the most powerful content formalism that is possible within an AsTRa implementation. This remark also characterizes the relationship between SB-ONE and FOPC as implemented with KN-OTTER. All SB-ONE assumption contents of a view can be translated to FOPC. So, it is possible that SB-ONE knowledge can be integrated into FOPC reasoning. This will be useful when SB-ONE view contents are related to FOPC view contents. For instance, instantiations of concepts or roles may be contained in the FOPC part of a view knowledge base.

In order to allow this kind of mixed representation, KN-OTTER was enhanced to take SB-ONE knowledge into account within a refutation process. It was enabled to recognize situations in the resolution process in which a query to the SB-ONE part of a view may be beneficial [Zimmermann, 1994]. This mechanism was based on the FOPC equivalence patterns of SB-ONE constructs (cf. Table 6.1).

However, the kind of hybrid reasoning that is described above is quite limited and specific to the currently available content formalisms SB-ONE and FOPC. In principle, an AsTRa implementation may offer an arbitrary number of content

[5] The domain of this formula will be explained in Chapter 7.

formalisms. If one of them is FOPC, like in BGP-MS, one approach to generalized hybrid reasoning is to let type-internal FOPC reasoning process the whole assumption type knowledge base, i.e. FOPC assumption contents together with the translations of all other contents. However, in this case the input for FOPC reasoning could become unnecessarily large. Therefore, a better approach could be to generalize the procedure of BGP-MS. I.e., FOPC reasoning would have to be extended by a general component for integrating useful assumption contents of other formalisms. Other kinds of hybrid reasoning like the meta-processing approach of the expert system shell Babylon [Christaller *et al.*, 1992] are also imaginable.

6.3.3 BGP-MS Is an AsTRa System

In this short section, BGP-MS, as defined so far, is related to the formal definitions of an AsTRa user modeling knowledge base and an AsTRa system, which were given in Section 5.2.4.

1. A BGP-MS user modeling knowledge base contains a set of views (in the broader sense of the term) and their view knowledge bases, $\mathcal{V}_{BGP\text{-}MS}$, and a set of stereotypes, $\mathcal{S}_{BGP\text{-}MS}$. Views are identified with assumption types, and stereotypes obey the AsTRa stereotype definition. So, a BGP-MS knowledge base is an AsTRa knowledge base $\langle \mathcal{AT}, \mathcal{S} \rangle$, with $\mathcal{AT} := \mathcal{V}_{BGP\text{-}MS}$ and $\mathcal{S} := \mathcal{S}_{BGP\text{-}MS}$.

2. By default, BGP-MS allows two modality symbols to be employed in view labels (i.e., assumption type labels), namely B for beliefs and W for wants; further modalities may be added by the developer of a user modeling system. BGP-MS provides the logic-based content formalisms SB-ONE and FOPC. BGP-MS is able to manage multiple UMKBs (for details, see Section 7.3.3). However, as specified for AsTRa systems, at any time one of these UMKBs is the designated current UMKB, UMKB_c. The interface functions of BGP-MS can access only UMKB_c. In sum, BGP-MS is an AsTRa system $\langle \mathcal{UMKB}, \text{UMKB}_c, \mathcal{F}, \mathcal{MOD} \rangle$, with $\mathcal{F} := \{\text{SB-ONE}, \text{FOPC}\}$, and $\mathcal{MOD} :\supseteq \{B, W\}$.

6.4 Extended AsTRa in BGP-MS

6.4.1 Reasoning with Modal Logic

In order to implement the extended AsTRa framework in BGP-MS, a proof procedure for modal logic was needed that can deal with the multi-modal, multi-agent formulas of the AL^+ logic. According to [Reichgelt, 1989], there are mainly two approaches to constructing a theorem prover for modal logic, namely the *modal* and the *reified* approach. The modal approach is to build specialized theorem provers for modal logic. The reified approach is based on the observation that the model theory of modal logic, i.e. possible worlds semantics, can be formulated in first-order logic. Hence, it is possible to translate all modal formulas into first-order logic and use a first-order theorem prover. For BGP-MS, the latter approach was more appropriate. With KN-OTTER, there already was a first-order prover at hand, which had been equipped with an interface to BGP-MS for purposes of view-internal FOPC reasoning.

In the rest of this section, first the classical procedure for translating modal formulas into first-order logic is discussed. Second, a more recent method is introduced, which will turn out to be advantageous for the purposes of BGP-MS. Then, an algorithm is described that permits the translation of modal formula schemes; the integration of schemes is needed for processing general relationships between assumption types. Finally, it will be specified how the mentioned mechanisms lead to a proof procedure that makes use of KN-OTTER.

Relational Translation

Recall that possible worlds semantics is based on a set of possible worlds which are related by an accessibility relation R. Formulas are interpreted with respect to a possible world w. A modal formula $\Box\phi$ is true in a world w, if ϕ is true in all worlds that are accessible from w via the relation R. Moore [Moore, 1980; Moore, 1985] observed that this meaning of modal sentences can be described by sentences of first-order logic. This led to the idea that modal formulas can become objects of first-order logic (they can be "reified"). Then, one is able to reason about them, provided that some axioms describe the conditions under which modal object sentences are true.

Let $[\phi]$ be the reified version of the modal sentence ϕ. Then, for example, the truth of formulas $\Box\phi$ could be described in first-order logic as follows (cf. [Reichgelt, 1989]):

$$\forall w\ (\mathrm{true}(w, [\Box\phi]) \ \equiv\ \forall w'\ R(w, w') \rightarrow \mathrm{true}(w', [\phi]))$$

Note that the use of a distinct object language for modal formulas and of a dedicated truth predicate requires the formulation of axioms that also define the semantics of standard first-order connectives, like

$$\forall w\ (\mathrm{true}(w, [\phi \wedge \psi]) \ \equiv\ true(w, [\phi]) \wedge \mathrm{true}(w, [\psi])$$

However, this can be avoided. Note that the interpretation of a modal formula in a world w is based on the interpretation of non-modal literals (which are the basic elements of the modal logic language) in other worlds w'. Now, for a literal P, instead of saying $true(w, [P])$ the world variable could be added as additional *world argument* to P, yielding $P(w)$. The same can be done for logical terms.

Then, the meaning of the modal formula $\Box P$ with respect to a world w can be translated to

$$\forall w' \; R(w, w') \to P(w')$$

Based on this principle, a translation function $\pi(\phi, w)$ can be defined that transforms a modal sentence into first-order logic—relative to a possible world—by stepping recursively through ϕ and translating sub-formulas $\Box\psi$ according to the equation

$$\pi(\Box\psi, z) = \forall z' R(z, z') \to \pi(\psi, z')$$

In general, the evaluation of modal sentences starts at the actual world w_0. I.e., in order to compute the relational translation $\Pi_R(\phi)$ of a complex modal sentence ϕ, $\pi(\phi, w_0)$ is executed.

A specific problem occurs when dealing with logical terms. Function symbols, too, will get an additional world argument, because they can be differently interpreted in different worlds. In addition, in different worlds also the object domains which terms are referring to may be different[6]. Hence, the set that quantifiers are ranging over depends on the considered world. In order to simplify the situation, the domain of the translated formulas can be taken as union of all domains of the possible worlds structure. Then, however, a special predicate $exists(w, t)$ must be introduced that states if a term t (more precisely, the object denoted by t) exists in world w. As a simple example, the translation of a literal $P(c)$ is demonstrated:

$$\pi(P(c), z) = exists(z, c(z)) \wedge P(z, c(z))$$

Note that, if all domains in the possible worlds structure are assumed to be identical ("constant domains") and all symbols are assumed to be rigid, i.e. their interpretation does not vary, the translation of $P(c)$ simplifies to

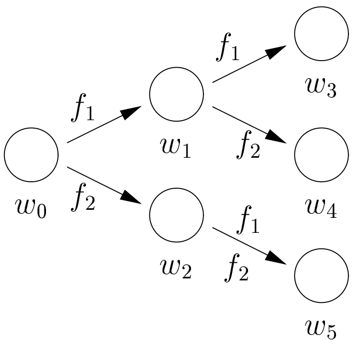
$$\pi(P(c), z) = P(z, c)$$

The kind of transformation of modal sentences into first-order logic that was described above does not pursue a truly reified approach, because it lacks the treatment of modal formulas as objects of the first-order language. We will say that such methods pursue a "translation approach". Since the above method uses a predicate R to explicitly represent the accessibility relation, it is called *relational translation*.

Functional Translation

Relational translation is a quite flexible method, but has one main disadvantage. The first-order formula that results from a translation is blown up with additional literals, particularly if the *exists* predicate is used. Theoretically, resolutions between R-literals in a resolution procedure are only needed to enable related

[6] In the literature, this is also called the "varying domains" problem [Ohlbach, 1991].



accessibility relation R	context access functions f_1, f_2
(w_0, w_1)	$f_1(w_0) = w_1$
(w_0, w_2)	$f_1(w_1) = w_3$
(w_1, w_3)	$f_1(w_2) = w_5$
(w_1, w_4)	
(w_2, w_5)	$f_2(w_0) = w_2$
	$f_2(w_1) = w_4$
	$f_2(w_2) = w_5$

Figure 6.4: Relational versus functional representation of possible world accessibility

resolution steps between normal literals. Since normal resolution procedures will not account for special predicates like R, too many useless resolution steps are possible.

There has been work on extending first-order provers to deal with modal logic, taking the opposite direction in comparison with the translation approach (e.g., see [Wallen, 1987; Jackson and Reichgelt, 1989]). A central idea of this approach is to associate with each formula and term a world index that indicates the evaluation context of formulas and terms. In the relational translation, there is a world variable for literals and terms, but the actual evaluation context, i.e. the world that is the value of this variable, is determined by a number of R-literals. It would be advantageous to completely describe the path from the actual world w_0 to an evaluation world by one complex term which would replace the world variable and make the R-literals superfluous.

This is the main idea of *functional translation*, which was invented by Ohlbach [Ohlbach, 1991]. This translation method "uses the fact that a binary relation can be represented by the domain-range relation of a set of one place functions" [Ohlbach, 1991, p.693]. That is, the accessibility relation R can be replaced by a set of *context access functions* that range over the set of possible worlds. This is illustrated in Figure 6.4, which is taken from [Ohlbach, 1991]. On the left, it shows a small set of possible worlds with accessibility links. The links immediately define the relation R, which is listed in the middle of the figure. On the right, a minimal set of context access functions is presented that can replace the accessibility relation, as the labels on the accessibility links demonstrate.

With context access functions, it is possible to describe paths through the possible worlds structure by function composition. In the example of Figure 6.4, the composition $f_1 \circ f_2$ describes the path from w_0 to w_4. Another possibility is to have *world terms* specify any world of the possible worlds structure. E.g.,

w_4 can be described by $f_2(f_1(w_0))$. Thus, in a possible worlds structure every world can be identified with a term that is constructed by sequential application of context access functions on the initial world w_0.

Analogous to relational translation, the crucial step of functional translation is the translation of formulas with modal operators. This is defined for the functional translation function $\pi_F(\phi, z)$, where z now is a world term, as follows (the $\forall$ quantifier refers to the "for all accessible worlds" of the possible worlds semantics):

$$\pi_F(\Box\psi, z) = \forall f \pi_F(\psi, f(z))$$

Thus, the translation function generates world terms while traversing modal operators. These world terms are in the end inserted as additional arguments into literals:

$$\pi_F(P, z) = P(z)$$

Logical terms are handled like in relational translation. Function symbols with varying interpretation also get a world term as additional argument, and an *exists* predicate is needed if there are varying domains in the possible worlds structure.

Note that a translated formula may contain terms $f(z)$ where f is quantified. At first sight, this seems to be a second order construct. Actually, it is not second order, since context access functions do not range over the object domain of the modal frame, but range over the distinct set of possible worlds. Still, the syntax is a second order syntax, which cannot be processed by theorem provers like OTTER. Ohlbach therefore recommends to introduce an 'apply' function to construct world terms. This function has two arguments, a context access function and a world term. The translation of modal operators is then redefined as

$$\pi_F(\Box\psi, z) = \forall f \pi_F(\psi, apply(f, z))$$

The purpose of this section is to develop a proof procedure for the logic AL^+, which shall be used in BGP-MS to extend the expressive power of basic assumption type representation. AL^+ is a multi-modal, multi-agent logic. I.e., the modal operator $\Box$ can be indexed with a modality m and an agent a; e.g., $\Box_{(B,U)}$ is the belief operator for agent U. Hence, in a possible worlds structure for such a logic, there is a distinct accessibility relation $R_{(m,a)}$ for each modality-agent pair. In relational translation, this would lead to the introduction of two additional parameters of the R-predicate. In functional translation, two new argument places are added to context access functions. So, for the purposes of AL^+, the translation of modal operators needs to be redefined again:

$$\pi_F(\Box_{(m,a)}\psi, z) = \forall f\ \pi_F(\psi, apply(f, m, a, z))$$

A specific problem of functional translation arises if the accessibility relation of the possible worlds structure is not serial. In this case, there may be "dead end" worlds from which no other world can be accessed. Then, context access functions necessarily are partial functions, which are not defined for all worlds. Since standard predicate logic cannot handle partial functions, context access functions must become total functions also in the serial case. To this end, on the semantics side an artificial element (an additional world) can be introduced. Then, all arguments for which a function is not defined can be mapped to this element. On the syntax side, a predicate $end(w, m, a)$ can be introduced, which is true if w is a dead end world according to the accessibility relation of m and a, $R_{(m,a)}$. Then, the next redefinition of the translation step for modal operators is as follows:

$$\pi_F(\Box_{(m,a)}\psi, z) \;=\; \neg end(z, m, a) \rightarrow \forall f\; \pi_F(\psi, apply(f, m, a, z))$$

It is obvious that, similar to relational translation, this step generates additional literals, which originally were to be avoided by functional translation. Hence, for the purposes of BGP-MS it seems sensible to make a seriality assumption for the accessibility relations of every modality-agent pair. An important consequence of this decision results from the fact that seriality of an accessibility relation $R_{(m,a)}$ corresponds to the axiom D of modal logic, $\Box_{(m,a)}\Phi \rightarrow \neg\Box_{(m,a)}\neg\Phi$. That is: In BGP-MS, because of simplifying functional translation by assuming seriality for the accessibility relations of all operators, AL^+ is implemented as a KD modal logic. Note that also the specialized AsTRa mechanisms for reasoning with negative assumptions (see Section 5.4) were based on the D axiom. BGP-MS implements these mechanisms, as will be described in Section 6.5. Hence, on the whole, *the knowledge representation and reasoning facilities of BGP-MS implement a KD modal logic.* This has two benefits: first, negative assumptions can be treated in a satisfying way, and second, functional translation (and hence full AL^+ reasoning) is significantly simplified.

For BGP-MS, two further simplifying assumptions are made: First, all function symbols (i.e., also the modality and agent indices of the modal operator $\Box$) are assumed to be rigid, and the object domains of the possible worlds structure are assumed to be constant. In this case, function symbols will not get a world term argument, and the insertion of 'exists' literals can be avoided.

The functional translation $\Pi_F(\phi)$ of a modal formula ϕ is defined analogously to the relational translation: $\Pi_F(\phi) = \pi_F(\phi, w_0)$. Modal knowledge bases, i.e. sets of modal formulas KB $= \{\psi_i\}$, are translated by translating each formula: $\Pi_F(\mathrm{KB}) = \{\Pi_F(\psi_i)\}$.

Further-on, we will abbreviate world terms $apply(f, m, a, z)$ with $f_{(m,a)}(z)$. In the following example of a functional translation, this notation is already used:

$$\begin{aligned}
\phi &= \Box_{(B,S)}\Box_{(W,U)}p_1(\mathrm{a}, b) \rightarrow \Box_{(B,S)}\neg\Box_{(B,U)}p_2(\mathrm{a}, b)\\
\Pi_F(\phi) &= \forall g \forall f\; p_1(g_{(W,U)}(f_{(B,S)}(w_0)), \mathrm{a}, \mathrm{b})\\
&\quad \rightarrow \forall v \neg \forall u\; p_1(v_{(B,U)}(u_{(B,S)}(w_0)), \mathrm{a}, \mathrm{b})
\end{aligned}$$

In order to be a useful means for realizing modal reasoning, a translation method must be sound and complete. These properties were proven by Ohlbach for both relational and functional translation [Ohlbach, 1991]. For BGP-MS, functional translation is used. Therefore, Ohlbach's theorem for functional translation is presented (in a simplified formulation):

Theorem 3 ([Ohlbach, 1991]) *A modal formula ϕ is satisfiable in a modal interpretation iff its functional translation $\Pi_F(\phi)$ is satisfiable in the corresponding first-order interpretation.*

As a corollary, we get:

Corollary 8 *An AL^+ formula ϕ follows from an AL^+ knowledge base KB (formally: $KB \models \phi$), iff the translated formula follows from the translated knowledge base (formally: $\Pi_F(KB) \models \Pi_F(\phi)$).*

This is the foundation of using functional translation for the implementation of extended AsTRa mechanisms in BGP-MS.

Processing Modal Axioms with SCAN

Without further specification, a modal operator is just a K operator with modus ponens and necessitation rule as its characterizing axioms (cf. Section 3.4.1). Because of the assumptions that have been made to simplify functional translation, AL^+ operators are realized in BGP-MS as KD operators, i.e., with the axiom D holding for all of them. The standard way to define further properties of modal operators is to specify other modal axioms, like the axioms 4 and 5 that were presented in Section 3.4.1. These axioms normally correspond to properties of the accessibility relation. The axiom D corresponds to seriality. Axiom 4, which in case of a belief operator expresses the capability of positive introspection, corresponds to transitivity. In order to do modal reasoning with the translation approach, it is necessary to express such properties in first-order logic and add them to the set of formulas under consideration.

In case of relational translation, the R predicate can immediately be used to express properties of the accessibility relation. For example, transitivity is expressed by the first-order formula

$$\forall x, y, z\, R(x, y) \wedge R(y, z) \rightarrow R(x, z)$$

For all standard modal axioms, there are such first-order correspondences. [van Benthem, 1984] discusses the relation between modal axioms and properties of the possible worlds structure in detail.

In case of functional translation, expressing correspondence properties is not that straightforward. Of course, for the most important axioms, correspondences were already found. Again, transitivity is taken as example:

$$\forall w\, \forall f, g\, \exists h\, (g(f(w)) = h(w))$$

This equation expresses that for every two-step world path there is a one-step path leading to the same world, which exactly is what transitivity means.

In BGP-MS, the developer of a user modeling system shall be able to freely specify the axioms that he wants to be considered by full AL^+ reasoning. Therefore, a procedure was needed that computes the FOPC correspondences of modal axioms automatically. first-order correspondences In fact, such a procedure exists, namely the SCAN[7] algorithm [Gabbay and Ohlbach, 1992]. Its main task is to compute the properties of the possible worlds structure that cause a given axiom to hold. Its main idea is to try to refute the negated and then FOPC-translated axiom with resolution. An axiom can be translated by treating formula variables as normal predicate symbols. Then, resolution is applied to the clause form of the translated axiom. This step is successful, if some clauses remain unresolved; they are implied by the negated axiom. Hence, by simply taking the contraposition of this implication[8], the negation of the remaining clauses imply the axiom. Therefore, the resulting first-order correspondence is generated by negating the remaining clauses, after transforming them back into a first-order formula first. Particularly in case of functional translation, it is important to apply a modified resolution procedure that conserves the conditions which made a resolution step possible. Finally, these conditions result in equations like the one that was given above as correspondence for transitivity.

As an example, the SCAN translation of a complex modal axiom is presented that permits SBUB implications to be considered by inferences that deal with SBUW assumptions:

$$\Big(\Box_{(B,S)}\Box_{(W,U)}\,\Phi \wedge \Box_{(B,S)}\Box_{(B,U)}\,(\Phi \rightarrow \Psi)\Big) \rightarrow \Box_{(B,S)}\Box_{(W,U)}\,\Psi \tag{6.2}$$

represents the rule that users want any implication of their immediate goals if they know the implication relation. From this modal formula scheme, SCAN generates the following conjunction of two equations:

$$\begin{aligned} &\forall w\,\forall q, r\,\exists f, g, t, u \\ &\quad g_{(W,U)}(f_{(B,S)}(w)) = u_{(B,U)}(t_{(B,S)}(w)) \\ &\quad \wedge\; u_{(B,U)}(t_{(B,S)}(w)) = r_{(W,U)}(q_{(B,S)}(w)) \end{aligned} \tag{6.3}$$

In all SCAN-generated equations, a universally quantified variable w takes the place of the initial world constant w_0, which usually appears in functionally translated formulas. Thus, the equations are applicable to arbitrary world paths, which reflects the fact that formula variables in modal schemes may be instantiated by arbitrary modal formulas.

SCAN will successfully compute the standard properties of the accessibility relation from the usual axioms for modal operators. Also schemes that do not

[7] Synthesize Correspondence Axioms for Normal logics

[8] The contraposition of an implication $a \rightarrow b$ is $\neg b \rightarrow \neg a$.

primarily aim at defining a property of the accessibility relation, but are to describe a general inference rule in the given logic, may become subject to SCAN processing—an example was presented above. However, SCAN does not always succeed. [van Benthem, 1984] states that there are axioms which have no first-order correspondence. Obviously, SCAN must fail on these axioms. But there are also cases where a first-order correspondence may exist, but SCAN (more exactly, the resolution procedure within SCAN) will not terminate. Hence, SCAN is not complete, but sound. [Gabbay and Ohlbach, 1992] proved that, if SCAN returns a result, then this is the desired correspondence.

Theorem 4 (Gabbay and Ohlbach) *If SCAN terminates for a formula α, then α is logically equivalent to SCAN(α).*

From this theorem and corollary 8, we obtain that SCAN and functional translation provide a sound and complete means for AL^+ reasoning.

Corollary 9 *An AL^+ formula ϕ follows from an AL^+ knowledge base $KB^{AL^+} := \mathcal{MF} \uplus \mathcal{MA}$, where $\mathcal{MF}$ is a set of modal formulas and $\mathcal{MA} := \{ma_j\}$ is a set of modal axioms such that SCAN(ma_j) exists, if and only if its translation follows from the union of the translated sets $\Pi_F(\mathcal{MF})$ and SCAN($\mathcal{MA}$) $:= \{SCAN(ma_j)\}$. Formally, this can be written as*

$$KB^{AL^+} \models \phi \iff \Pi_F(\mathcal{MF}) \cup SCAN(\mathcal{MA}) \models \Pi_F(\phi)$$

This means that, based on implementations of Π_F and SCAN, a first-order theorem prover can be employed for inferences with an AL^+ knowledge base including modal axioms.

Modal Reasoning with Functional Translation and KN-OTTER

With the functional translation function Π_F and the SCAN procedure, the necessary means for implementing inference procedures for the AL^+ logic with formula schemes are available. Both Π_F and SCAN were implemented for BGP-MS. While functional translation did not cause any problems worth mentioning, the implementation of SCAN was more difficult. In particular, the modified resolution procedure of SCAN had to be realized. For this purpose, KN-OTTER was applied to slightly modified input formulas. This modification was suggested in [Gabbay and Ohlbach, 1992] for enabling standard resolution theorem provers like OTTER to simulate the modified resolution procedure. The conception and realization of the implementation of both functions is documented in detail in [Schreck, 1995; Simon, 1995].

Besides being employed within the SCAN implementation, KN-OTTER can also be utilized to do AL^+-level reasoning. In Section 5.3.1, an extended AsTRa implementation was required to provide functions AL^+-*derivable*, AL^+-*forward* and AL^+-*consistent* as foundation of the AL^+ reasoning level. With functional translation, the SCAN algorithm, and the *kn-otter* function for FOPC reasoning at hand, such functions can be realized. Quite naturally, the specifications for these functions will be very similar to those of the access functions of the FOPC content formalism. Here, *kn-otter* is applied to a FOPC-translated AL^+ knowledge base instead of a FOPC assumption type knowledge base. Remember that an AL^+ knowledge base contains modal formulas and modal axioms, i.e. $\mathrm{KB}^{AL^+} = \mathcal{MF} \uplus \mathcal{MA}$. Then:

- AL^+-*consistent*$(\mathrm{KB}^{AL^+}, \phi) :=$
  $\mathrm{not}(\textit{kn-otter}(\Pi_F(\mathcal{MF}) \cup \mathrm{SCAN}(\mathcal{MA}), \{\Pi_F(\phi)\}))$

- AL^+-*derivable*$(\mathrm{KB}^{AL^+}, \phi) := \textit{kn-otter}(\Pi_F(\mathcal{MF}) \cup \mathrm{SCAN}(\mathcal{MA}), \{\neg\Pi_F(\phi)\})$

As with the FOPC content formalism reasoning function $\textit{forward}_{FOPC}$, also the AL^+ forward reasoning function AL^+-*forward* works analogously to its corresponding *consistent* function, and it also makes use of the extended version of *kn-otter* that returns derived resolvents, *kn-otter** (cf. Section 6.3.2).

- AL^+-*forward*$(\mathrm{KB}^{AL^+}, \phi) := \textit{kn-otter*}(\Pi_F(\mathcal{MF}) \cup \mathrm{SCAN}(\mathcal{MA}), \{\Pi_F(\phi)\})$

6.4.2 Combining Partitions and Modal Reasoning

Modal Reasoning without Partitions?

In the last section, mechanisms for reasoning with the multi-modal, multi-agent logic AL^+ were presented. In Section 5.3.1, it was pointed out that both type-internal and type-external expressions of the extended AsTRa framework can be represented in this logic. Hence, in principle it would be possible to represent the whole UMKB of BGP-MS in AL^+ and do all UMKB reasoning using the above-mentioned methods. In order to imitate the built-in inferences of the partition mechanism, it would be necessary to explicitly add hierarchy axioms to the set of modal axioms, $\mathcal{MA}$, of the UMKB. They can be processed with SCAN. The first and most straightforward way to implement this "modal logic only" approach to UMKB reasoning would be to maintain the whole BGP-MS UMKB not as a set of modal formulas and axioms, but as one big set of translated modal expressions, $\Pi_F(\mathcal{MF}) \cup \mathrm{SCAN}(\mathcal{MA})$. This set is exactly what *kn-otter* needs as input in AL^+ reasoning functions .

In Section 5.3.1, however, it was argued further that an extended AsTRa implementation should offer both type-external and type-internal reasoning. With AL^+ representation and reasoning only, a kind of type-internal reasoning could theoretically be realized by picking out all assumptions of one type from the set of translated formulas and then apply *kn-otter* to this subset only (which would be a set of AL formulas). In principle, it is possible to do so; all assumptions of a given type have an identical translation pattern. However, the real advantage of type-internal reasoning, namely the use of specific content formalisms for specific needs, would be lost. And also if specialized content formalisms are not employed (e.g., if only FOPC is used for representing assumption contents), it is still a little easier to process a set of FOPC formulas than to process the set of their corresponding AL formulas (cf. the argumentation in Section 5.3.1).

In BGP-MS, there even is one more reason for retaining type-internal (i.e., *view-internal* in terms of BGP-MS) reasoning in combination with the partition mechanism for representing assumption types. Hierarchy axioms can simulate the partition inferences inheritance and propagation in AL^+ reasoning, but their translations are complex and will probably slow down KN-OTTER processing notably. It is much more efficient to employ the specialized procedures of the partition mechanism for partition inheritance and propagation.

Since both the use of specific content formalisms for view-internal UMKB contents and the built-in inferences of the BGP-MS partition mechanism had proven useful for user modeling, they were to be preserved. Therefore, it became necessary to look for a solution that uses both partition mechanism and content formalisms, and that smoothly integrates the use of translation techniques for modal reasoning with those basic facilities.

It is straightforward to determine that view-external expressions (modal formulas and modal axioms) will be translated using the standard procedures and kept aside of the partition hierarchy. The more problematic issue is: How to deal with view-internal expressions? A first approach is to store view-internal expressions both in the partition hierarchy and, in form of translated AL correspondences, together with the translations of view-external expressions. This would result in having the whole UMKB FOPC-translated as discussed above, but now in parallel to the partition hierarchy with its contents. For view-external reasoning, the translated UMKB could be used. View-internal reasoning would not be influenced at all.

However, this approach seems to cause a waste of space. Not only are the translations of view-internal expressions redundant; moreover, due to partition inheritance more than one translation would have to be stored for contents of non-leaf partitions. A content expression ac that is stored in a non-leaf partition V is contained in a number of views, $V_1, \ldots, V_n$. Thus, it represents the view-internal expressions[9] $V_1{:}ac, \ldots, V_n{:}ac$. These expressions correspond to the AL formulas $\mathcal{M}(V_1)\, ac, \ldots, \mathcal{M}(V_n)\, ac$. So, n formulas would have to be translated for one partition content expression. Furthermore, the translated UMKB would have to be updated not only because of new entries into the user model, but also because of upward propagation of partition contents. So, keeping both the translations of the whole UMKB and the partition hierarchy in parallel is space-intensive and causes additional maintenance problems.

[9] Recall that in BGP-MS views are identified with assumption types.

Modal Reasoning with Partitions!

In this section, an alternative approach will be described that avoids the disadvantages discussed above (see also [Pohl, 1996a]). It is based on the simple fact that, for all assumption contents ac of a view V, the corresponding AL formulas have the same form, namely $\mathcal{M}(V)\,ac$. This implies that also their functional translations follow an identical pattern. Thus, a very efficient way of translating view-internal expressions can be developed. Because of its efficiency, it permits translation of view-internal expressions on demand (i.e., when translations are needed for view-external reasoning), which relieves modal logic representation and reasoning mechanisms from dealing with partition inferences.

An example will illustrate the above considerations. The view-internal expression SBUW:printed(userdoc) corresponds to the modal formula

$$\Box_{(B,S)}\Box_{(W,U)}\,\mathrm{printed}(\mathrm{userdoc})$$

which is translated into

$$\forall f, g\ \mathrm{printed}(g_{(W,U)}(f_{(B,S)}(\mathrm{w}_0)), \mathrm{userdoc})$$

In general, all expressions of the form

$$\Box_{(B,S)}\Box_{(W,U)}\,\phi$$

will be translated into

$$\forall f, g\ \phi[g_{(W,U)}(f_{(B,S)}(\mathrm{w}_0))]$$

where $\phi[t]$ means adding a first argument term t to all atoms in ϕ. It is easy to see that both the quantifier list $\forall f, g$ and the world term $g_{(W,U)}(f_{(B,S)}(\mathrm{w}_0))$ can be generated directly from the modal operator sequence $\Box_{(B,S)}\Box_{(W,U)}$ and hence from the corresponding view label SBUW. This observation can be generalized: For a view V, a world term $wt(V)$ as well as a quantifier list $Q(V)$ can be generated. Then, computing the functional translation of a view-internal expression $V{:}ac$ means translating its corresponding AL expression $\mathcal{M}(V)\,ac$ and therefore results in $Q(V)\,ac[wt(V)]$.

World term and quantifier list are precisely defined as follows[10]:

Definition 6.3 (world term, quantifier list) *Let $f^1, \ldots, f^n$ be pairwise different function symbols. For a view $V = a_1m_1 \ldots a_nm_n$, the* world term *of V, $wt(V)$, is defined*

$$wt(V) := f^n_{(m_n,a_n)}(\ldots(f^1_{(m_1,a_1)}(w_0))\ldots)$$

The quantifier list *of V, $Q(V)$, is defined*

$$Q(V) := \forall f^1 \ldots \forall f^n$$

[10] Note that negative assumption types are not yet considered.

These are the prerequisites for defining a specialized translation method T for view-internal expressions $V{:}ac$.

Definition 6.4 (functional translation of view-internal expressions) *The functional translation of a view-internal expression, $T(V{:}ac)$, is defined as follows:*

$$T(V{:}ac) := Q(V)\, ac[wt(V)] = \Pi_F(\mathcal{M}(V)\, ac)$$

In BGP-MS, world term $wt(V)$ and quantifier list $Q(V)$ are computed once for each view V and stored along with the base partition P_V in the partition hierarchy. Generating $T(V{:}ac)$ then consists of the following steps:

1. From ac and $wt(V)$, generate $ac[wt(V)]$.
2. Put $Q(V)$ in front of the result.

The first step can be simplified further: Note that for the view-internal expressions $V_1 : ac, \ldots, V_n : ac$, where all of $V_1, \ldots, V_n$ have a common direct or indirect superpartition P (or are equal to P), the content expression ac is stored only in P. Along with ac we can store its "translation frame" $ac[\,]$, which is the view-independent part of $ac[wt(V)]$, as sequence of formula substrings[11]. An example: For ac = printed(userdoc) we obtain $ac[\,]$ = (`"printed(" ",userdoc)"`). Since $ac[\,]$ is view-independent, it needs to be computed only once and stored at no more places in the partition hierarchy than ac itself. Now, generating $ac[wt(V)]$ simply means inserting $wt(V)$between each two adjacent substrings of $ac[\,]$. Thus, we have a very efficient method of generating the translation

$$T(V{:}ac) \;=\; Q(V)\, ac[wt(V)]$$

of an expression ac contained in view V. E.g., for translating the view-internal expression SBUW:printed(userdoc), the following ingredients are needed:

1. the world term $wt(\mathrm{SBUW}) = f^2_{(W,U)}(f^1_{(B,S)}(w_0))$, which in BGP-MS is internally encoded as the string `"apply(f2,W,U,apply(f1,B,S,w0))"`;
2. the quantifier list $Q(\mathrm{SBUW}) = \forall f^1 f^2$ = `"all f1 all f2"`;
3. and the translation frame printed(userdoc)$[\,]$ = (`"printed(" ",userdoc)"`)

Then, the translation

$$T(\textsf{SBUW:printed(userdoc)}) := Q(\mathrm{SBUW})\textsf{printed(userdoc)}[wt(\mathrm{SBUW})]$$

results in

```
"all f1 all f2 printed(apply(f2,W,U,apply(f1,B,S,w0)),userdoc)"
```

[11] Recall that we identify an assumption content ac with its FOPC translation $\tau(ac)$. Taken precisely, the translation frame of an assumption content ac is $\tau(ac)[\,]$.

This is a string representation of

$$\forall f^1 f^2 \mathrm{printed}(f^2_{(W,U)}(f^1_{(B,S)}(w_0)), \mathrm{userdoc})$$

which is exactly what would result from $\Pi_F(\mathcal{M}(\mathrm{SBUW})\,\mathrm{printed}(\mathrm{userdoc}))$.

Finally, the translation of a complete view can be defined as the set

$$T(V) \;:=\; \{T(V{:}ac) \mid ac \in \mathrm{KB}_V\}$$

Thus, AL^+ reasoning, which is demanded to involve both view-internal and view-external user model contents, can be performed in BGP-MS using the permanently stored functional translations of view-external expressions together with translations of all views of the current partition hierarchy. KN-OTTER can be applied to process the resulting formula set. A crucial property of the presented approach to view-external reasoning is that the mechanisms of view-internal reasoning remain unchanged.

Modal Reasoning in BGP-MS

In the previous section, an optimization of the functional translation function for view-internal formulas was described. Now, the main functions for modal reasoning in BGP-MS can be defined. Following the extended AsTRa specification, AL^+-*derivable*, AL^+-*forward*, and AL^+-*consistent* functions are needed. In Section 6.4.1, a possible implementation of these functions was sketched. There, it was specified that each function needs to be applied to an AL^+ knowledge base (in case of an AsTRa system: UMKB^{AL^+}), which is translated into FOPC with functional translation and SCAN. In the previous subsection, it was indicated that there are more efficient ways to compute this translation from the BGP-MS UMKB directly, without the detour of generating UMKB^{AL^+}. In this subsection, the direct translation of the UMKB, $T(\mathrm{UMKB})$, is defined.

As a prerequisite, the notion of a BGP-MS user modeling knowledge base needs to be redefined. Moreover, the AsTRa database access functions **store** and **fetch** will have to be slightly (re-)defined for BGP-MS in order to take the needs of the translation function T into account. Then, T can be defined for the whole UMKB. The definition will make clear that $T(\mathrm{UMKB})$ corresponds to collecting the pre-computed translations of view-external formulas and modal axioms, in addition to forming the translations of view-internal expressions.

Definition 6.5 (View) *A View is the BGP-MS realization of an AsTRa assumption type. Hence, a view is a pair $\langle V, KB_V\rangle$ with V being the view label and KB_V being the view knowledge base. The view knowledge base KB_V is a set of pairs, each consisting of an assumption content ac and its translation frame $ac[\,]$:*
$KB_V := \{\langle ac_1, ac_1[\,]\rangle, \ldots, \langle ac_n, ac_n[\,]\rangle\}$

Note that when implemented, content formalisms may use specialized storage mechanisms within partitions instead of forming views as sets of pairs. However, it is demanded that each assumption content can be handled individually and that it be linked to its translation frame.

Definition 6.6 (BGP-MS User Modeling Knowledge Base)
A BGP-MS user modeling knowledge base UMKB is an extended AsTRa knowledge base $\langle \mathcal{AT}, \mathcal{S}, \mathcal{MF}, \mathcal{MA} \rangle$ *(cf. Definition 5.19) such that*

- $\mathcal{AT}$*, the set of assumption types, is a set* $\{V_i\}$ *of views .*
- $\mathcal{S}$ *is a set of BGP-MS stereotypes.*
- $\mathcal{MF}$*, the set of modal formulas, is a set of pairs* $\langle \phi, \phi_T \rangle$ *with* ϕ *being a view-external* AL^+ *formula and* ϕ_T *being its functional translation* $\pi_F(\phi)$*.*
- $\mathcal{MA}$*, the set of modal axioms, is a set of pairs* $\langle ma, ma_T \rangle$ *with* ma *being an* AL^+ *axiom and* ma_T *being its SCAN translation* $SCAN(ma)$*.*

In Section 5.3.1, the database access functions **store** and **fetch** were defined to take AL^+ expressions into account. Both functions distinguish whether their input expression is a modal axiom, a type-external modal formula, or a simple AL formula that corresponds to a type-internal expression. Modal axioms and type-external formulas are added to and retrieved from the sets $\mathcal{MA}$ and $\mathcal{MF}$, respectively. AL formulas $\mathcal{M}(AT)\,ac$ are handled like type-internal expressions $AT{:}ac$ by applying type-internal content functions $store_{F(ac)}$ and $fetch_{F(ac)}$. However, an extended BGP-MS UMKB differs from an extended AsTRa UMKB in maintaining the functional translations along with UMKB contents. Therefore, **store** and **fetch** and also the $store_F$ and $fetch_F$ functions of content formalisms F must be redefined. All of them will have to take care of the specific structure of a BGP-MS UMKB.

First, the specification of the content formalism access functions $store_F$ and $fetch_F$ functions (cf. Section 5.2.3) is adapted to the BGP-MS implementation of the extended AsTRa framework.

- $store_F(\mathrm{KB}_V, ac) := (\mathrm{KB}_V := \mathrm{KB}_V \cup \{\langle ac, ac[\,]\rangle\})$
- $fetch_F(\mathrm{KB}_V, ac) := \langle ac, ac[\,]\rangle \in \mathrm{KB}_V$

Note that this specification is theoretically motivated. Actual implementations of content formalisms will probably use specific methods to add expressions to and retrieve expressions from a knowledge base. However, they must take care of $ac[\,]$ being generated and/or linked to the internal representation of ac in the view knowledge base KB_V.

Now, also the specification of extended **store** and **fetch** from Section 5.3.1 can be adapted as follows:

- **store**(UMKB, ϕ) :=
$$\begin{cases} store_{F(ac)}(\mathrm{KB}_V, ac) & \phi = V{:}ac \\ store_{F(ac)}(\mathrm{KB}_V, ac) & \phi = \mathcal{M}(V)\, ac \in AL \\ \mathcal{MF} := \mathcal{MF} \cup \{\langle \phi, \phi_T \rangle\} & \phi \text{ is an } AL^+ \text{ formula } \notin AL \\ \mathcal{MA} := \mathcal{MA} \cup \{\langle ma, ma_T \rangle)\} & \phi \text{ is an } AL^+ \text{ axiom and} \\ & \quad \text{SCAN}(\phi) \text{ returns with } ma_T \end{cases}$$

- **fetch**(UMKB, ϕ) := $\begin{cases} fetch_{F(ac)}(\mathrm{KB}_V, ac) & \phi = V{:}ac \\ fetch_{F(ac)}(\mathrm{KB}_V, ac) & \phi = \mathcal{M}(V)\, ac \\ \langle \phi, \ldots \rangle \in \mathcal{MF} & \phi \text{ is an } AL^+ \text{ formula } \notin AL \\ \langle \phi, \ldots \rangle \in \mathcal{MA} & \phi \text{ is an } AL^+ \text{ axiom} \end{cases}$

Now, the translation of a UMKB, $T(\mathrm{UMKB})$, can be defined:

Definition 6.7 (FOPC translation of a BGP-MS UMKB)
The FOPC translation of a BGP-MS UMKB, $T(UMKB)$, is defined as

$$T(UMKB) := T(\mathcal{AT}) \cup T(\mathcal{MF}) \cup T(\mathcal{MA})$$

with

- $T(\mathcal{AT}) := \bigcup_{V \in \mathcal{AT}} T(V)$
- $T(\mathcal{MF}) := \{\phi_T \mid \langle \phi, \phi_T \rangle \in \mathcal{MF}\}$
- $T(\mathcal{MA}) := \{ma_T \mid \langle ma, ma_T \rangle \in \mathcal{MA}\}$

In Section 6.4.1, AL^+-*derivable*, AL^+-*forward*, and AL^+-*consistent* functions were defined according to the requirements of the extended AsTRa framework. They take an AL^+ knowledge base as input, translate it into FOPC and invoke the function *kn-otter* to process it. Since the translation function T can be applied directly to the UMKB, the intermediate step of using the AL^+ version of the UMKB is no longer necessary. So, genuine AL^+ reasoning functions are not needed for realizing the AL^+ reasoning level in BGP-MS. They can be replaced by functions *e-derivable*, *e-forward*, and *e-ask* ("e" for "extended") that are applied to the UMKB directly. However, in their use of *kn-otter* they work analogous to the AL^+ functions:

- *e-derivable*(UMKB, ϕ) := *kn-otter*(T(UMKB), $\{\Pi_F(\neg\phi)\}$)
- *e-forward*(UMKB, ϕ) := *kn-otter**(T(UMKB), $\{\Pi_F(\phi)\}$)
- *e-consistent*(UMKB, ϕ) := not(*kn-otter*(T(UMKB), $\{\Pi_F(\phi)\}$))

Based on these functions, the specification of the implementation of the AL^+ reasoning level in BGP-MS can be completed. The AL^+ level reasoning functions **derivable**$_{AL^+}$, **forward**$_{AL^+}$, and **consistent**$_{AL^+}$ make use of the above BGP-MS-specific "e-" functions:

- $\mathbf{derivable}_{AL^+}(\text{UMKB}, \phi) := \textit{e-derivable}(\text{UMKB}, \phi)$
- $\mathbf{forward}_{AL^+}(\text{UMKB}, \phi) := \textit{e-forward}(\text{UMKB}, \phi)$
- $\mathbf{consistent}_{AL^+}(\text{UMKB}, \phi) := \textit{e-consistent}(\text{UMKB}, \phi)$

6.5 Negative Assumption Types in BGP-MS

In the AsTRa framework, negative assumption types were introduced for a specialized handling of negative assumptions. Like other assumption types, negative assumption types are containers of assumption content expressions, possibly represented with different content formalisms. However, the use of negative types deviates from the use of the standard positive types. The main reason is that inferences within the assumption type knowledge base are not sound in case of negative types. But this does not mean that reasoning is not possible with negative types. In Section 5.4.3, it was shown that a negative assumption is related to other assumptions in the UMKB, negative and positive ones. These relationships are independent of the assumption content, so that three kinds of relations between assumption types were established, namely forward derivation, backward derivation, and inconsistency relations. They are exploited by specific negative assumption type reasoning mechanisms that are used at two additional AsTRa reasoning levels.

In BGP-MS, a negative assumption type is implemented like all assumption types: as a partition with a partition knowledge base. In principle, such a "negative partition" does not differ from other, "positive" partitions. It can become linked into the partition hierarchy, and hence also a stereotype partition may be associated with a negative partition. Therefore, we will further-on also refer to negative views.

All type-internal AsTRa reasoning levels take negative assumption types into account. In BGP-MS, all type-internal reasoning levels, including the levels NA and $TI + NA$ for special negative assumption type reasoning (cf. Section 5.4.3), were implemented according to the AsTRa specifications. All type-internal levels are based on the knowledge base access functions of content formalisms. For BGP-MS, we described the access functions of both available content formalisms SB-ONE and FOPC in Sections 6.3.1 and 6.3.2, respectively. In addition, the relations 'derivable', 'forward', and 'inconsistent' are needed that range over the set of assumption types, i.e. over the set of views in BGP-MS. Every assumption type is in 'derivable', 'forward', or 'inconsistent' relation to only a finite (and typically small) subset of the theoretically infinite set of possible types. Hence, for every assumption type that is defined to be part of a BGP-MS UMKB, its possible 'derivable', 'forward', and 'inconsistent' relationships can be exhaustively computed at the time of definition. This information is then stored along with the base partition of the assumption type.

Example 6.1: With the definition of the negative assumption type SB∼UB the following sets are computed (cf. Example 5.5 in Section 5.4):

- derivable(SB∼UB) = $\{\langle \text{SBUB}, \neg \rangle\}$
- forward(SB∼UB) = $\emptyset$
- inconsistent(SB∼UB) = $\{\langle \text{SBUB}, \varepsilon \rangle\}$

Thus, $\mathbf{derivable}_{NA}(UMKB, \text{SB}{\sim}\text{UB}{:}ac)$ will look for SBUB:$\neg ac$ in order to derive the query expression, $\mathbf{forward}_{NA}(\text{UMKB}, \text{SB}{\sim}\text{UB}{:}ac)$ will not infer anything, and $\mathbf{consistent}_{NA}(\text{UMKB}, \text{SB}{\sim}\text{UB}{:}ac)$ will look for SBUB:ac in order to find a contradicting UMKB content.

This example indicates that negative assumption type reasoning is limited in case of SB-ONE. Reconsider that $\mathbf{derivable}_{NA}(\text{UMKB}, \text{SB}{\sim}\text{UB}{:}ac)$ looks for $\neg ac$ in view SBUB, i.e., it needs to negate the given assumption content ac. However, SB-ONE has no negation operator that would correspond to the FOPC negation operator $\neg$ which is employed in the specifications of derivable(AT), forward(AT), and inconsistent(AT). Recall that these sets consist of pairs $\langle AT', sig \rangle$, with AT' being an assumption type and $sig \in \{\neg, \varepsilon\}$. Only pairs with $sig = \varepsilon$ can be considered for negative type reasoning in case of SB-ONE.

So, view-internal reasoning copes with negative assumption types. The next step is to look at view-external reasoning. Negative assumptions can be smoothly integrated into the view-external mechanisms. A negative assumption corresponds to a formula of $AL^{\neg}$, an extension to the assumption logic AL that additionally permits negations in front of modal operators. Obviously, $AL^{\neg}$ is a subset of AL^{+}, so that, theoretically, negative assumptions can be processed using AL^{+} reasoning techniques. Practically, in Section 6.4.2 we defined extended reasoning mechanisms that integrate view-internal and view-external contents of a BGP-MS user modeling knowledge base. These mechanisms are based on procedures for translating modal formulas into first-order logic. A translation function T was defined that particularly for view-internal contents realizes a specialized and efficient procedure for generating the needed translations.

In this translation procedure, contents of negative views can be treated like other view-internal knowledge. Consider the following example: The view-internal expression

$$\mathsf{SB{\sim}UW{:}printed(userdoc)}$$

corresponds to the modal formula

$$\Box_{(B,S)} \neg \Box_{(W,U)} \text{printed}(\text{userdoc})$$

which is translated into

$$\forall f \neg \forall g \ \text{printed}(g_{(W,U)}(f_{(B,S)}(\text{w}_0)), \text{userdoc})$$

In general, all expressions of the form

$$\Box_{(B,S)} \neg \Box_{(W,U)}\ \phi$$

will be translated into

$$\forall f \neg \forall g\ \phi[g_{(W,U)}(f_{(B,S)}(\mathrm{w}_0))]$$

A similar observation can be made for all possible negative views. That is, there is a world term $wt(V)$ and a quantifier list $Q(V)$ also for a negative view V. World term and quantifier list can be used together with the translation frames $ac[\,]$ of view contents ac to generate the translations of all view-internal expressions $V{:}ac$. In order to cover negative views, we must redefine the formation of world terms and quantifier lists slightly.

Definition 6.8 (world term, quantifier list) *Let $f^1, \ldots, f^r$ be pairwise different function symbols. For a view $V = \nu_1 a_1 m_1 \ldots \nu_r a_r m_r$ (cf. Definition 5.21), the world term of V, $wt(V)$, is defined*

$$wt(V) := f^r_{(m_r,a_r)}(\ldots(f^1_{(m_1,a_1)}(w_0))\ldots)$$

The quantifier list of V, $Q(V)$, is defined

$$Q(V) := t(\nu_1)\forall f^1 \ldots t(\nu_r)\forall f^r$$

where $t(\nu_i)$ transforms the negation symbol $\sim$ of view labels into the logical negation symbol $\neg$. I.e., $t(\nu_i) := \neg$ for $\nu_i = \sim$, and $t(\nu_i) := \varepsilon$ for $\nu_i = \varepsilon$.

In sum, negative assumption types are implemented as negative views. Negative views can be used analogously to normal, positive views. They provide a means to represent negation of modal operators in a specialized way. In BGP-MS, negative views should be used instead of modal reasoning, if the need for negation of modal operators is the only representation need that cannot be satisfied by view-internal mechanisms. However, negative views can also be easily integrated into AL^+ reasoning as it is implemented in BGP-MS.

6.6 Exploiting the Flexibility of AsTRa in BGP-MS

The AsTRa framework was designed for flexibility concerning the use of its representation and reasoning facilities. A basic AsTRa system offers a great degree of freedom in setting up the set of modalities, assumption types, and stereotypes. For inferring assumptions that are implicit in the user modeling knowledge base, forward reasoning and/or backward deduction can be employed. In an extended

AsTRa system, expressive power is enhanced, but the flexibility of a basic AsTRa system is not lost, since all basic mechanisms, i.e. assumption types, stereotypes, and content formalisms, are still available. There is always the choice to use either specialized, type-internal mechanisms or powerful, type-external reasoning.

The AsTRa framework encourages the modular development of a user modeling shell. The most important modules of a basic AsTRa system are its content formalisms. In a specific system installation, it should be possible to include a subset of the available content formalisms only. In principle, an AsTRa implementation could also allow the developer of a user modeling system to add one or more further content formalisms; such formalisms would have to satisfy (at least) the practical requirements of a logic-based formalism. In an extended system with negative types, it is left to the developer to decide how to represent negative assumptions: with or without negative types. For systems without further needs, negative types may become the only extension to basic AsTRa facilities.

In this section, we will describe how the flexibility of the AsTRa framework was transferred into BGP-MS.

6.6.1 Flexible Use of Content Formalisms

In BGP-MS, two formalisms for representing assumption contents are available: the terminological representation system SB-ONE, and FOPC based on the automated theorem prover OTTER. Both formalisms can be used in parallel, and FOPC reasoning is able to take conceptual knowledge of SB-ONE into account.

Of course, it is possible to use only one of these formalisms. Obviously, if only expressions of $\mathcal{L}_{SB\text{-}ONE}$ are used as assumption contents ac in type-internal expressions $AT{:}ac$, then only SB-ONE structures will be established in the partitions of BGP-MS. If only FOPC formulas are employed, partitions will only contain these formulas. The choice between SB-ONE and FOPC can be different for every assumption type. For example, in a concrete application it may be sensible to use conceptual structures for assumptions concerning beliefs, and to use logical expressions for assumptions about user goals.

However, the choice of content formalisms is not only possible during the use of BGP-MS. If the user model developer decides a priori that merely SB-ONE is needed for user model representation, then it is possible to configure a version of BGP-MS with SB-ONE as the only content formalism. In this case, further representational facilities of SB-ONE can be utilized. In case of combined use of SB-ONE and FOPC, there is a clear separation between both formalisms. SB-ONE is to represent terminologies, i.e. hierarchies of concepts with attributes. FOPC is to represent rule-like knowledge, but also simple assertions. So, in the FOPC part of a view, assertions concerning concepts and attributes of the SB-ONE part can occur. In terms of KL-ONE languages, FOPC can be considered an *ABox* formalism for SB-ONE. The separation of terminological and assertional knowledge proved advantageous for hybrid FOPC/SB-ONE reasoning; for a detailed discussion see [Zimmermann, 1994]. Therefore, the SB-ONE facilities for representing basic assertions were not employed. However, when SB-ONE is used alone, then the *individual concepts* and *individual roles* of SB-ONE can be employed for assertion representation. Then, representation of assertions will be limited to the one-place and two-place predicates that correspond to SB-ONE concepts and roles.

In case of exclusive use of SB-ONE, the language of SB-ONE, $\mathcal{L}_{SB\text{-}ONE}$, can be extended. Table 6.4 shows the two additional language elements that can be used as assumption content expressions. It extends Table 6.1, which defined the terminological part of $\mathcal{L}_{SB\text{-}ONE}$. C is a concept variable, R is a role variable, and o, o_1, o_2 may be replaced by constant symbols. On the left, there are the additional $\mathcal{L}_{SB\text{-}ONE}$ expressions, the middle column shows the corresponding standard notation of description logics, and the right column contains FOPC translations.

$\mathcal{L}_{SB\text{-}ONE}$	*descr. logic*	*FOPC translation*
$(C\ o)$	$o \in C$	$C(o)$
$(R\ o_1\ o_2)$	$(o_1, o_2) \in R$	$R(o_1, o_2)$

Table 6.4: Assertional $\mathcal{L}_{SB\text{-}ONE}$ expressions and their FOPC translations

6.6.2 Flexible Use of Representation and Reasoning

There are two possible extensions to a basic AsTRa system: full AL^+ reasoning as a general approach to represent and reason about user modeling knowledge that goes beyond the capabilities of assumption types and contents, and negative assumption types for specialized handling of negative assumptions. In Section 6.5, we have shown that the BGP-MS implementation of negative assumption types, negative views, can be neatly integrated with AL^+ reasoning. Thus, it is possible to use both extensions together. Still, each of them can be employed without the other one. On the one hand, it may well be the case that negative assumptions are not of special interest to a user modeling system, while other kinds of extended knowledge are made use of, e.g. view-external inference rules for user model acquisition. There is a simple solution for this case: The user model developer refrains from defining negative assumption types and from using reasoning levels NA and $TI + NA$. On the other hand, several applications of BGP-MS have already demonstrated that there is a need for negative assumptions in systems that do not require the use of other extended mechanisms. In this case, a version of BGP-MS can be configured that does not employ modal reasoning, but offers negative assumption types and the respective reasoning levels exclusively.

The AsTRa specification says that modal reasoning is to be offered as an additional option only. The reason for not employing modal logic as exclusive representation and reasoning system is that the flexibility of type-internal AsTRa mechanisms, particularly the specialized capabilities of content formalisms, shall not be lost. The AsTRa framework specifies four non-modal reasoning levels and one modal logic level. All these levels are implemented in BGP-MS, so that in principle, a BGP-MS user could freely choose between them.

Recall the definitions of the global access functions **tell** and **ask** (cf. Definition 5.13); their first parameter allows one to choose a reasoning level:

1. $\mathbf{tell}(L, \mathrm{UMKB}, AT{:}ac) :=$
   if $\mathbf{consistent}_L(\mathrm{UMKB}, AT{:}ac)$ then
   $\mathbf{store}(\mathrm{UMKB}, AT{:}ac)$ and $\mathbf{forward}_L(\mathrm{UMKB}, AT{:}ac)$

2. $\mathbf{ask}(L, \mathrm{UMKB}, AT{:}ac) :=$
   $\mathbf{fetch}(\mathbf{UMKB}, AT{:}ac)$ or $\mathbf{derivable}_L(\mathrm{UMKB}, AT{:}ac)$

In these two definitions, all AsTRa reasoning functions are involved. In case of **tell**, both $\mathbf{consistent}_L$ and $\mathbf{forward}_L$ work at the same level. In BGP-MS, a slightly different and even more flexible approach is taken. In accordance with the AsTRa specification, there are two global UMKB access functions, called **bgp-ms-tell** and **bgp-ms-ask**. Both accept a view-internal expression or an AL^+ formula as input or query expression. In addition, it is possible to separately specify the reasoning level for each involved reasoning function. This means that in case of **bgp-ms-tell**, different levels can be specified for consistency checking and forward derivation. If nothing is specified, the default reasoning level 0 is employed for all reasoning functions. That is, without further specification **bgp-ms-tell** and **bgp-ms-ask** are simple entry and retrieval functions.

A further, very powerful means of determining the behavior of BGP-MS reasoning processes is the use of AL^+ formula schemes. Formula schemes can be seen as axioms that define the meaning of modal operators and hence determine possible inferences. In the AL^+ implementation of BGP-MS, and also at the negative assumption type reasoning levels, every modal operator is a KD operator, with the classical modal axioms K and D holding (the axioms are listed in Section 3.4.1). The K axiom holds in any normal modal logic with possible worlds semantics. The D axiom holds due to the seriality assumption concerning the accessibility relations of all AL^+ operators, which was made in order to optimize the process of translating the full meaning of view-internal UMKB contents into first-order logic. In addition, D is presumed for negative type reasoning. The user model developer can add further modal axioms to a BGP-MS UMKB in order to influence the behavior of modal operators or to describe general relationships between assumption types. The only restriction is that the SCAN algorithm must be able to translate a new axiom, i.e. it must have a first-order correspondence.

6.6.3 Flexible Use of Reasoning Directions

The two main reasoning facilities of an AsTRa system, like of most logic-based reasoning systems, are forward reasoning in combination with a new entry to the UMKB, and backward reasoning in combination with a query to the UMKB (cf. Section 5.2.4). Both ways of reasoning are employed to discover secondary assumptions about the user that are implicit in the UMKB; both have their advantages and disadvantages. A backward reasoning process is goal-directed because it is focused on answering a query, while a forward reasoning process has no goal and can get lost in the space of possibly implicit assumptions. However, forward reasoning can be decoupled from the activities of a user modeling application, since an application will normally not have to wait for a reaction of the user modeling system after a call to **tell**. In opposition, an application will normally wait for the answer to a query. In a typical user modeling situation, an application queries the user model when it wants to individualize its subsequent behavior. I.e., the subsequent behavior of the application depends on the answer, and the application will wait. In his system GUMAC, Kass applies only forward reasoning to process a considerable number of user model acquisition rules [Kass, 1991]. However, he considers this a suboptimal procedure and suggests a combination of limited forward and backward reasoning as a superior method (cf. Section 4.1.4).

In BGP-MS, both forward and backward reasoning are available. Both mechanisms can be applied in a flexible manner. For its **bgp-ms-tell** function, BGP-MS offers forward reasoning as a choice, and when calling the **bgp-ms-ask** function, the application can decide if backward reasoning is performed or not. The choices are possible because separate **store**, **fetch**, **forward**, and **derivable** functions are available. Depending on what is determined by the application, the appropriate function is invoked. Thus, the application can decide whether there is time to wait for an answer or not, or whether it is willing to share computing resources with the forward reasoning process of the user modeling system or not. Like Kass, we recommend to use both forward and backward reasoning. Forward reasoning will possibly pre-compute some of the implicit assumptions that a later query might request, and backward reasoning will take care that no implicit assumption is missed.

6.6.4 Accessing the UMKB: Bottom-Up and Top-Down

An AsTRa system mainly pursues a bottom-up approach to integrating type-internal knowledge with type-external, modal logic knowledge. Its basic mechanism is assumption type representation, with a set of formalisms for assumption contents. These facilities are extended with techniques for representing and reasoning with modal formulas; because of their modal logic semantics, type-internal contents can be integrated into modal logic reasoning processes. In contrast, [Kobsa, 1992] suggested a top-down approach, where modal logic, respectively first-order translations of modal formulas were proposed as central formalism. In this approach, as many translated modal formulas as possible are internally represented by partitions and content formalisms, since these mechanisms are specialized and hence more efficient in interesting cases. In Section 5.3.1, it was suggested for an extended AsTRa system that the modal logic AL^+ (not translated AL^+—the system must be able to translate formulas automatically, if needed) can optionally be used as overall interface language for UMKB contents. In this case, following the ideas of [Kobsa, 1992], the system will determine the optimal representation for a modal formula by itself. If the formula is an element of the assumption type logic AL, i.e., if it is of the form $\mathcal{M}(AT)\,ac$, it can be associated with an assumption type AT. Then, for the first-order part ac of such a formula, the appropriate content formalism $F(ac)$ will be determined (cf. Section 5.3.1).

BGP-MS allows both bottom-up and top-down access to its UMKB. On the one hand, the standard way to formulate a type-internal expression is to separate assumption type and assumption content information, like in our notation AT:ac. AT is an assumption type label, and ac can be an expression of either $\mathcal{L}_{SB\text{-}ONE}$ or $\mathcal{L}_{FOPC}$. Such an expression is represented internally as SB-ONE or FOPC expression in the assumption type knowledge base KB_{AT}. On the other hand, also $AL^{\neg}$ formulas $\mathcal{M}(AT)\,ac$ can be used for type-internal expressions. Hence, if an AL^+ formula ϕ is used as input or query expression, BGP-MS attempts to detect whether ϕ is a (type-internal) $AL^{\neg}$ formula, i.e., whether $\phi = \mathcal{M}(AT)\,ac$. If so, AT is determined as the appropriate assumption type. Now, the problem occurs that $t_{SB\text{-}ONE}(\mathcal{L}_{SB\text{-}ONE})$ and $t_{FOPC}(\mathcal{L}_{FOPC})$ overlap – FOPC is the target language of all t_F functions, and t_{FOPC} is the identity function. In Section 5.3.1, a preference ordering was suggested as a solution to this kind of problem. According to the idea of the top-down approach, namely to use the most specific formalism possible, SB-ONE is preferred to FOPC. The first-order part ac of an $AL^{\neg}$ formula $\mathcal{M}(AT)\,ac$ is always a FOPC formula. If it is a translation of an SB-ONE construct, then the corresponding SB-ONE construct is used as assumption content. For instance, the formula

$$\Box_{(B,S)}\Box_{(B,U)}\ \forall x\ \mathrm{whale}(x) \rightarrow \mathrm{fish}(x)$$

will be identified with a view-internal expression

SBUB:(isa whale fish)

However, if ac does not correspond to an SB-ONE expression, it will be represented as FOPC assumption content. For instance,

$$\Box_{(B,S)}\Box_{(W,U)}\ \forall x\ \mathrm{shark}(x) \wedge \neg\mathrm{looks}(x, \mathrm{friendly}) \rightarrow \mathrm{attacks}(\mathrm{flipper}, x)$$

just corresponds to

SBUW: all x shark(x) & –looks(x,friendly) -> attacks(flipper,x)

Chapter 7

User Modeling with BGP-MS

In the life of a user modeling system two main phases can be distinguished. First, at *development time*, the user model structure for the application domain is built, domain knowledge may be integrated into the user modeling component, and rules and procedures for user model acquisition are defined or implemented. Second, during *run time*, the user modeling system will exchange information about the user with the application and dynamically construct and maintain the user model.

User modeling shell systems should support both phases. They should provide well-defined interfaces that can be used by the developers of a user modeling system to set up initial knowledge about the domain(s) of the adaptive system. They should also provide well-defined interfaces for the communication between user modeling systems on the one hand and other participants in the user modeling process like adaptive applications (user model consumers in the terms of [Brajnik and Tasso, 1994; Kay, 1995]) and user-information providers (sensors in the terms of [Orwant, 1995]) on the other hand.

Both development time and run time communication interfaces are closely related to the representation and reasoning facilities of a user modeling shell. Most of the work to be done in the development phase is passing knowledge to the representation system, which has to ensure the consistency of domain knowledge and stereotypes. During run time, reasoning mechanisms acquire assumptions about the user from communicated observations or from primary assumptions, and they may be used in answering queries about the current contents of the user model.

In Chapter 6, the representation and reasoning capabilities of the user modeling shell system BGP-MS were discussed, but still on a fairly abstract level. This chapter is aimed at illustrating both the operation and the use of these capabilities in detail. For this purpose, a (fictitious) application scenario is employed, so that the reader shall get a good and concrete impression of how user model representation and reasoning in BGP-MS works. In Section 7.1, development time aspects will be discussed, while Section 7.2 will deal with run time user modeling.

Finally, Section 7.3 presents components of BGP-MS that are not genuine parts of the representation and reasoning facilities of BGP-MS, but are related to them in several ways. It will we described how BGP-MS supports user model acquisition and maintains information about the sources of BGP-MS UMKB contents. Furthermore, we will discuss the notion of a *domain*, which was introduced to BGP-MS as a way of structuring user modeling knowledge on a meta-level, and which is useful when both multiple users and multiple applications, user model consumers, and user-information providers shall be handled by a BGP-MS user modeling server. Communication mechanisms that are prerequisite for server activities will be described last.

BGP-MS has been implemented in Common Lisp. All UMKB access mechanisms are therefore available as Lisp functions. Hence, the straightforward way of utilizing BGP-MS is to develop an application in Lisp that integrates BGP-MS as a submodule. However, BGP-MS can also run as a separate process from applications. In this case, applications will communicate with BGP-MS using its inter-process communication interface, i.e., using inter-process messages to access BGP-MS. During development time, the standard way of using BGP-MS is to employ its functional interface within a Lisp environment. In addition, graphical interfaces for defining partition hierarchies and for setting up stereotypes are available, which however will not be described here (see [Kobsa and Pohl, 1995]). During run time, a user modeling application will typically communicate with BGP-MS via its inter-process message interface. Function calls and messages are almost identical; messages will invoke the namesake functions. The only difference is that the Lisp quote character ', which must be used in functions to prevent expressions from being evaluated, needs not be used in messages. In the following, function names will be printed in **bold** letters, while message names will be printed in a sans serif font for distinction purposes.

7.1 Development Time

In the development phase of a user modeling system, the developer first needs to specify what kinds of knowledge the user modeling knowledge base will contain. Candidates are all kinds of assumptions about the user, system domain knowledge, stereotypes, user model acquisition rules, etc. Furthermore, parts of the UMKB already have to be filled with pre-defined contents, with the exception of assumptions about individual users. Hence, in BGP-MS development activities concerning the UMKB consist of two steps: First, the structure of the UMKB is set up by introducing assumption types and by setting up the assumption type hierarchy. In addition, stereotypes can be defined. The range of possible assumption types is determined by the set of permissible modalities, which can be extended by the developer. Second, assumption contents and other UMKB contents may be pre-defined where necessary. Typically, system domain knowledge of type SB is engineered, and pre-defined stereotyped assumptions are filled into the different stereotype partitions. In addition, extended user modeling knowledge like global acquisition rules, perhaps in the form of modal axioms, may be established in the knowledge base. In the following subsections, the representation and reasoning facilities of BGP-MS will be illustrated insomuch as they are related to development time. We assume that the developer has a full version of BGP-MS at hand, which provides all facilities that were described in Chapter 6.

7.1.1 Example Scenario

For illustrating the representation and reasoning features of BGP-MS, we assume a somewhat futuristic scenario. Nevertheless, it is realistic in the sense that it deals with information filtering and adaptation to user preferences, which are relevant topics for user modeling systems.

Imagine a modern home entertainment environment with interactive television. Nowadays, interactive television lets the spectator choose between a fixed set of alternatives in presentation, like a number of different camera positions in broadcasts of sports or other events. There have also been experiments on having the spectator choose between various endings of a TV movie. Our example is more sophisticated: the TV environment attempts to adapt emissions to characteristics of individual spectators, i.e., to its individual users. For this purpose, it employs BGP-MS to maintain a user modeling knowledge base that contains assumptions about user beliefs and preferences. It makes use of the user model to adapt utterances of actors to the assumed knowledge of the user about concepts or facts of the domain, and to accommodate events and actions according to what the user is assumed to prefer to happen.

In the specific examples that are given below, the user (the spectator) is watching a broadcast of the "Flipper" TV series, which was originally produced in the 1960s in the U.S.A. The hero of this series is Flipper, a half-tame dolphin. The author watched broadcasts of this series in German television in the early 1970s. There, he learned that dolphins are able to attack and possibly kill sharks, which at least in every second episode appeared dangerously. In the following, this fact will play a crucial role.

7.1.2 Assumption Types

A BGP-MS user, i.e. the developer of a user modeling system who is working with BGP-MS, should first be concerned with the modalities that are going to be used by the application under consideration. In BGP-MS, the standard set of modalities is $\{B, W\}$; the developer can add further symbols to this set. In our example, assumptions about user preferences are to be modeled. So, the developer uses

```
(add-modality 'Pref)
```

to add the symbol 'Pref' to the modality set. Of course, there is no inherent semantics in this symbol, but we assume that the developer will use this modality in connection with user preferences.

Now, it is time to define the set of assumption types and possible hierarchical type relations. Any assumption type label that obeys Definition 5.1 and is in accordance with the set of modalities is permissible. For instance,

```
(define-assumption-type 'SBUB)
```

defines the SBUB assumption type; since it is the first type defined, any sub- or superordination relations to other types cannot be set up yet.

In our example, we also need the definitions

```
(define-assumption-type 'SB~UB)

(define-assumption-type 'SBUPref)
```

that introduce the negative type SB∼UB and the type SBUPref, which makes use of the additional modality.

BGP-MS implements assumption types as views in a partition hierarchy. For every assumption type a partition is created. It constitutes the base partition of the view, which implements the new assumption type. With every assumption type, i.e., with every view V, BGP-MS associates several bits of information, which are computed at definition time and then stored along with the corresponding partition. First, there are the sets derivable(V), forward(V), and inconsistent(V) that are required for reasoning with negative assumption types. Second, there are the world term $wt(V)$ and the quantifier list $Q(V)$ that are needed for AL^+ reasoning, namely for translating view-internal UMKB contents into FOPC. Table 7.1 lists this type-related information for the three assumption types defined so far. Note that $wt(V)$ and $Q(V)$ are not given in the formal notation of Chapter 6 but as character strings that can immediately become part of $\mathcal{L}_{FOPC}$ expressions. Such character strings can be easily filled into the slots of the translation frames $ac[\,]$ of assumption contents ac when generating the FOPC translations $T(V{:}ac)$ of view-internal expressions. The reason is that translation frames are implemented as lists of character strings (cf. Section 6.4.2). Such a list can be considered as one string with "gaps" for world terms and quantifier lists.

So far, all assumption types defined are unrelated to others. The next assumption type definitions are

```
(define-assumption-type 'SB)

(define-assumption-type 'SBMB :subtypes '(SB SBUB))
```

SBUB: derivable(SBUB) = $\emptyset$
forward(SBUB) = $\{\langle \text{SB}{\sim}\text{UB}, \neg\rangle\}$
inconsistent(SBUB) = $\{\langle \text{SB}{\sim}\text{UB}, \varepsilon\rangle\}$
wt(SBUB) = `"apply(f2,B,U,apply(f1,B,S,w0))"`
Q(SBUB) = `"all f1 all f2"`

SB∼UB: derivable(SB∼UB) = $\{\langle \text{SBUB}, \neg\rangle\}$
forward(SB∼UB) = $\emptyset$
inconsistent(SB∼UB) = $\{\langle \text{SBUB}, \varepsilon\rangle\}$
wt(SBUB) = `"apply(f2,B,U,apply(f1,B,S,w0))"`
Q(SBUB) = `"all f1 -all f2"`

SBUPref: derivable(SBUPref) = $\emptyset$
forward(SBUPref) = $\{\langle \text{SB}{\sim}\text{UPref}, \neg\rangle\}$
inconsistent(SBUPref) = $\{\langle \text{SB}{\sim}\text{UPref}, \varepsilon\rangle\}$
wt(SBUPref) = `"apply(f2,Pref,U,apply(f1,B,S,w0))"`
Q(SBUPref) = `"all f1 all f2"`

Table 7.1: Information associated with assumption types in BGP-MS

They introduce the SB type and SBMB as common supertype of SB and SBUB. To represent these type relationships, BGP-MS introduces hierarchy links from both partition SB and partition SBUB to partition SBMB.

The set of assumption types is not very specific to the application that we have in mind. On the contrary, stereotypes usually are domain-specific. In our example, there will be two stereotypes: One of them is of a more general character and is supposed to contain assumptions about the preferences of spectators that like action-oriented films (i.e., SBUPref assumptions). The other one is very specific to the example scenario and shall be filled with assumptions about mainly terminological knowledge of advanced spectators of “Flipper” (i.e., SBUB assumptions). The necessary definitions are[1]

```
(define-stereotype 'action :types '(SBUPref)
  :activation-condition ...
  :retraction-condition ...
  )

(define-stereotype 'fl-adv :types '(SBUB)
```

[1] Stereotype definitions in BGP-MS include specifications of conditions for stereotype activation and retraction. We will not deal with this issue here; see [Kobsa and Pohl, 1995] for details. Only one activation condition is specified, which will be referred to in Section 7.3.1.

```
    :activation-condition
      '(bgp-ms-ask (B S (fl-adv *current-user*)))
    :retraction-condition ...
    )
```

Thus, a stereotype 'action' with one stereotype partition SBactionPref, and a stereotype 'fl-adv' (short for "advanced spectator of 'Flipper' ") with the only stereotype partition SBfl-advB are established.

Figure 7.1 summarizes the definitions of this section. It shows the partitions and the hierarchy links that BGP-MS constructed according to the specifications of the developer. The dashed arrows represent potential links to stereotype partitions.

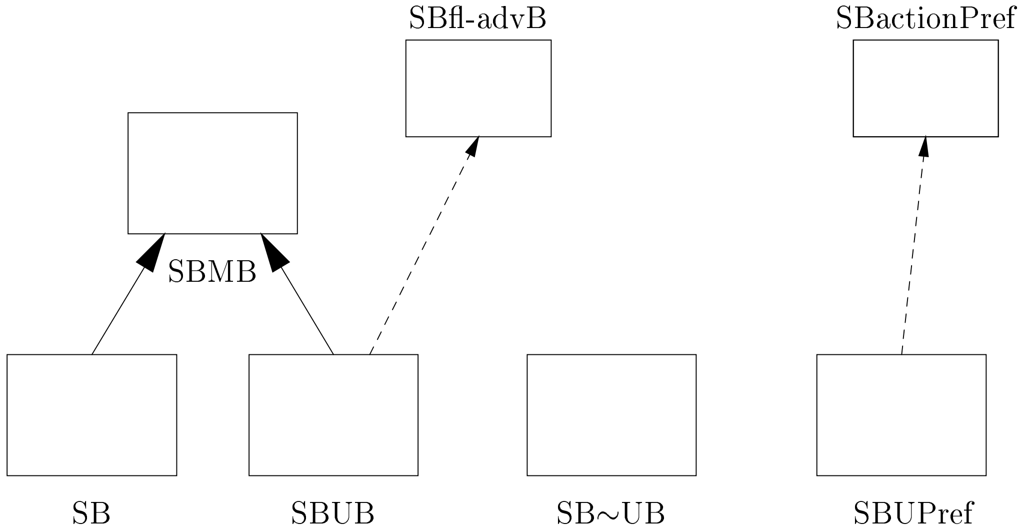
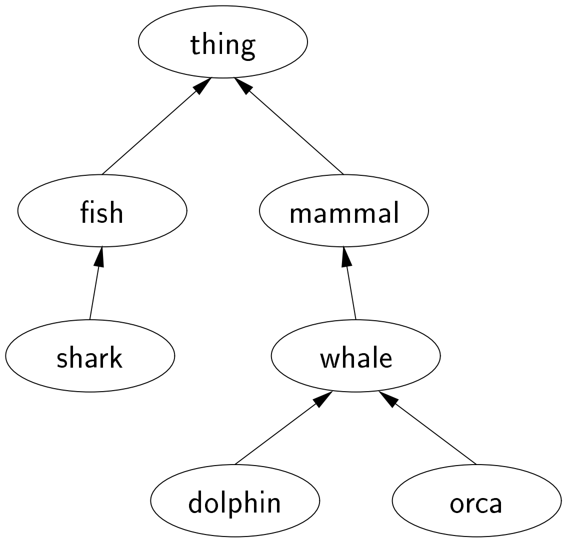


Figure 7.1: Partitions in the "Flipper" UMKB

7.1.3 Assumption Contents

After assumption types and stereotypes have been set up, a-priori knowledge can be entered into the UMKB. Typically, system domain knowledge, stereotyped assumptions, and inference rules for user model acquisition are determined beforehand.

BGP-MS provides two UMKB access functions, namely **bgp-ms-tell** and **bgp-ms-ask**. They can be used to employ the UMKB access functions of the different AsTRa reasoning levels, all of which are available in BGP-MS. For this purpose, both functions allow for parameter options that control the level of reasoning for each reasoning activity, i.e. consistency checking and forward derivation in case of **bgp-ms-tell**, and backward derivation in case of **bgp-ms-ask**.



(:concept fish)
(:concept mammal)
(:isa shark fish)
(:isa whale mammal)
(:isa dolphin whale)
(:isa orca whale)

Figure 7.2: Domain knowledge (assumption type SB): concept taxonomy

The default behavior of both functions is not to invoke any reasoning processes (i.e., to use reasoning level 0). This default behavior is usually sufficient when **bgp-ms-tell** is used during development time for filling the UMKB with a-priori knowledge. Consistency checking might be used in case the user model developer is not completely certain whether he is employing consistent a-priori knowledge or not. Forward reasoning might be invoked to discover implicit contents in system knowledge or in stereotypes. However, experience with BGP-MS has shown that particularly the latter feature has not been in demand of developers so far. In the following development time examples, **bgp-ms-tell** will be used without any parameter options in order to simply store a-priori entries into the UMKB. **bgp-ms-ask** and the possible parameter options of both **bgp-ms-tell** and **bgp-ms-ask** will be explained later on when they are used during run time of the example scenario.

At first, examples of the use of SB-ONE are presented. A taxonomy of marine beings is established in the SB partition. Parts of this taxonomy, together with further classifications concerning the character of some of the species, are assumed to be believed by advanced "Flipper" spectators. So, SB-ONE is also employed for adding stereotyped assumptions into the SBfl-advB stereotype partition.

Figure 7.2 shows an excerpt of the taxonomy that the developer enters into the UMKB as SB assumption contents. On the left, the typical graphical notation of KL-ONE systems is used. Ovals denote concepts, and arrows denote concept inheritance, i.e. IS-A links between concepts. On the right, the corresponding $\mathcal{L}_{SB\text{-}ONE}$ expressions are listed. The developer can use the BGP-MS interface variant of the $AT{:}ac$ notation for type-internal expressions to enter such conceptual hierarchies. For example,

```
(bgp-ms-tell '(SB (:concept fish)))
```

enters the SB-ONE concept fish into partition SB, as direct subconcept of the generic thing concept that is present in all partitions. **bgp-ms-tell** is applied to one argument that denotes a new content for the UMKB. In this case, the notation for type-internal expressions is utilized: A list with the assumption type as first element and the assumption content as second element[2]. IS-A relations between concepts in SB are defined similarly:

```
(bgp-ms-tell '(SB (:isa shark fish)))
```

adds the concept shark into SB as subconcept of fish (shark $\sqsubseteq$ fish, in terms of description logic). With a number of analogous calls to **bgp-ms-tell**, the whole taxonomy is constructed in SB.

From a theoretical point of view, stereotypes do not constitute new assumption types. They consist of a number of knowledge bases (in BGP-MS: a number of partitions), each of which is associated with an assumption type. However, in BGP-MS stereotype partitions are labeled similar to assumption types, and they are handled like assumption types as far as the definition of stereotype contents is concerned. Thus, **bgp-ms-tell** can be used to fill stereotype partitions like it is used to add contents to assumption type partitions. For example,

```
(bgp-ms-tell '(SBfl-advB (:isa shark dangerous)))
```

states that advanced spectators of "Flipper" are assumed to know that sharks are dangerous. Here, dangerous is a concept which is to denote the class of all dangerous objects (marine beings, in this case).

For the user modeling knowledge base of our example, the second content formalism of BGP-MS, first-order predicate calculus (FOPC), is also employed. First, a simple fact is added to the SBfl-advB stereotype partition, namely that the predicate hero holds for the object that is denoted by the constant flipper:

```
(bgp-ms-tell '(SBfl-advB "hero(flipper)"))
```

Note that expressions of $\mathcal{L}_{FOPC}$ have to be put in double quotes in order to become better readable to BGP-MS. Although hero is a one-place predicate, there is no namesake concept among the SB-ONE structures in the SBfl-advB partition. This is not necessary, since FOPC contents are independent of SB-ONE contents. However, logical expressions may involve predicates that correspond to SB-ONE concepts. In this case, the FOPC inference engine KN-OTTER will attempt to take related SB-ONE contents into account for drawing hybrid inferences. An example of a more complex FOPC content formula, which involves a predicate with an associated SB-ONE concept, is the rule

$$\forall x, y\ \mathrm{dangerous}(x) \land \mathrm{hero}(y) \rightarrow \mathrm{attacks}(y, x) \tag{7.1}$$

[2]For implementational reasons, the terminal symbols of $\mathcal{L}_{SB\text{-}ONE}$, concept, isa, and, and all are prefixed with a colon, thereby becoming special keyword symbols of Lisp.

(read: "Dangerous objects will be attacked by heroes."). The developer considers this rule a preference rule, i.e., the user may perhaps be assumed to prefer that this rule is obeyed. Thus, it belongs to assumption type SBUPref. The developer enters it into the partition of the 'action' stereotype that is associated with SBUPref, namely SBactionPref:

```
(bgp-ms-tell
  '(SBactionPref
    "all x y dangerous(x) & hero(y) -> attacks(y,x)"))
```

7.1.4 Extended Contents

The above preference rule that was ascribed to the potential group of action-interested users contained the predicate dangerous, which is related to the namesake SB-ONE concept. However, this concept is an assumption content of type SBUB, since it is contained in the SBUB-associated partition of the 'fl-adv' stereotype. Thus, facts that satisfy the first premise of the preference rule (7.1) will probably also be assumption contents of type SBUB. Moreover, facts about heroes will typically be SBUB assumptions. When the 'action' stereotype is activated, the contents of SBactionPref will become SBUPref contents via an inheritance link from partition SBUPref to partition SBactionPref. Type-internal inferences within the SBUPref view could take advantage of rule (7.1), but the rule premises will probably never be satisfied within SBUPref.

The problem is that a preference rule may need SBUB assumptions that satisfy its premises in order to infer its conclusion. So, a first solution to the problem is to replace the type-internal preference rule by the type-external AL^+ formula

$$\forall x, y\, \Box_{(B,S)}\Box_{(B,U)}(\text{dangerous}(x) \wedge \text{hero}(y)) \rightarrow \Box_{(B,S)}\Box_{(Pref,U)}\text{attacks}(y, x)$$

For entering AL^+ formulas into a BGP-MS UMKB, a Lisp interface syntax has to be used, which is defined in Table 7.2. Its main characteristic is that modal expressions $\Box_{(m,a)}\phi$ are formulated as a three-element structure ($m\ a\ \phi$). Moreover, as usual in Lisp, a prefix notation is used, i.e., all connectives, predicates and function symbols are first elements of subexpression lists.

AL^+ formulas can be added to the UMKB with **bgp-ms-tell**:

```
(bgp-ms-tell
  '(all x y (-> (B S (B U (and (dangerous x) (hero y))))
                (B S (Pref U (attacks y x))))))
```

If such a call were made, BGP-MS would recognize that this is no simple $AL^{\neg}$ formula and therefore cannot be handled like a type-internal expression. Hence, the call would result in the formula along with its functional FOPC translation being added to the set of modal formulas, $\mathcal{MF}$.

1. Lisp syntax for FOPC formulas:

```
<term>          ::= variable | constant | ( function <term>+ )
<formula>       ::= predicate | ( predicate <term>+ )
                    | ( and <formula>+ ) | ( or <formula>+ )
                    | ( -> <formula> <formula> )
                    | ( <-> <formula> <formula> )
                    | ( all | exists variable+ <formula> )
```

2. Extension for AL formulas (equivalent to view-internal expressions):

```
<AL-formula>    ::= ( modality agent <formula>)
                    | ( modality agent <AL-formula>)
```

3. Extension for AL^+ formulas:

```
<formula>       ::= ( modality agent <formula> )
<AL+-formula>   ::= <formula>
```

4. Extension for modal axioms in AL^+:

```
<formula>       ::= ( :var symbol )
```

Table 7.2: BGP-MS interface syntax of AL^+

However, such a specific rule now allows SBUB contents to satisfy its specific premises only. A less specific solution is to define an axiom that generally allows for deductions that combine SBUB assumptions with SBUPref-internal preference rules:

$$\Box_{(B,S)}\Box_{(B,U)}\Phi \wedge \Box_{(B,S)}\Box_{(Pref,U)}(\Phi \rightarrow \Psi) \;\rightarrow\; \Box_{(B,S)}\Box_{(Pref,U)}\Psi$$

In our example, the developer adds this axiom to the UMKB instead of the specific type-external rule above. The `(:var` *symbol*`)` expressions below represent formula variables (cf. the syntax for AL^+ axioms in Table 7.2).

```
(bgp-ms-tell
  '(-> (and (B S (B U (:var phi)))
            (B S (Pref U (-> (:var phi) (:var psi)))))
       (B S (Pref U (:var psi)))))
```

This axiom is quite similar to the example axiom (6.2) for which a SCAN translation was presented in Section 6.4.1. The only differences are that here, the modality Pref is used instead of W, and that the arrangement of modalities is slightly different. The SCAN translation of the above axiom can therefore be obtained from the translation example (6.3) by making the corresponding changes. The result is

$$\begin{aligned}
&\forall w\, \forall q, r\, \exists f, g, t, u \\
&\quad g_{(B,U)}(f_{(B,S)}(w)) = u_{(Pref,U)}(t_{(B,S)}(w)) \\
&\quad \wedge\;\; u_{(Pref,U)}(t_{(B,S)}(w)) = r_{(Pref,U)}(q_{(B,S)}(w))
\end{aligned}$$

Since the SCAN algorithm returns successfully, the axiom, paired together with its SCAN translation, is added to $\mathcal{MA}$. For the purposes of BGP-MS and its modal reasoning facilities, the SCAN translation is stored in a string notation that is made to fit the translations of view-internal expressions as generated by the translation function T:

```
"all w q r  exists f g t u
    apply(g,B,U,apply(f,B,S,w))
      = apply(u,Pref,U,apply(t,B,S,w))
  & apply(u,Pref,U,apply(t,B,S,w))
      = apply(r,Pref,U,apply(q,B,S,w))"
```

7.1.5 The UMKB after Development Time

In general, the result of the development process of BGP-MS is a UMKB that contains a priori definable user modeling knowledge. In this section, we will comprehensively describe the UMKB that was generated for our example application.

1. $\mathcal{AT}$ = { SB, SBUB, SBMB, SB∼UB, SBUPref }. The partition of view SBMB is superpartition of the SB and SBUB partitions. For all views, assumption type information (cf. Table 7.1) is computed and stored. So far, the view knowledge bases are:

   - KB_{SB} = { ⟨ (concept fish), ("all x fish(" ",x) -> thing(" ",x)") ⟩ ;
     ⟨ (isa shark fish), ("all x shark(" ",x) -> fish(" ",x)") ⟩ ;
     ...}
   - $\text{KB}_{SBUB} = \text{KB}_{SBMB} = \text{KB}_{SB\sim UB} = \text{KB}_{SBUPref} = \emptyset$

2. $\mathcal{S}$ = { action, fl-adv, ...}. Both relevant stereotypes have only one stereotype partition, i.e.

   action = { ⟨SBactionPref, SBUPref⟩ }, and
   fl-adv = { ⟨SBfl-advB, SBUB⟩ }

   Among others, the stereotype partitions contain the following assumptions:

   - $\text{KB}_{SBactionPref}$ =

     { ⟨ all x y dangerous(x) & hero(y) -> attacks(y,x),
     ("all x y (dangerous(" ",x) & hero(" ",y))
     -> attacks(" ",y,x)") ⟩ ;
     ...}

   - $\text{KB}_{SBfl\text{-}advB}$ =

     { ⟨ (isa shark dangerous),
     ("all x shark(" ",x) -> dangerous(" ",x)") ⟩ ;
     ⟨ hero(flipper) , ("hero(" ",flipper)") ⟩ ;
     ...}

3. $\mathcal{MF} = \emptyset$

4. $\mathcal{MA}$ = { ⟨ $\Box_{(B,S)}\Box_{(B,U)}\Phi \wedge \Box_{(B,S)}\Box_{(Pref,U)}(\Phi \rightarrow \Psi) \rightarrow \Box_{(B,S)}\Box_{(Pref,U)}\Psi$,

```
"all w q r exists f g t u
     apply(g,B,U,apply(f,B,S,w))
       = apply(u,Pref,U,apply(t,B,S,w))
   & apply(u,Pref,U,apply(t,B,S,w))
  = apply(r,Pref,U,apply(q,B,S,w))"
```
   ⟩ }

Table 7.3: A BGP-MS UMKB after development time

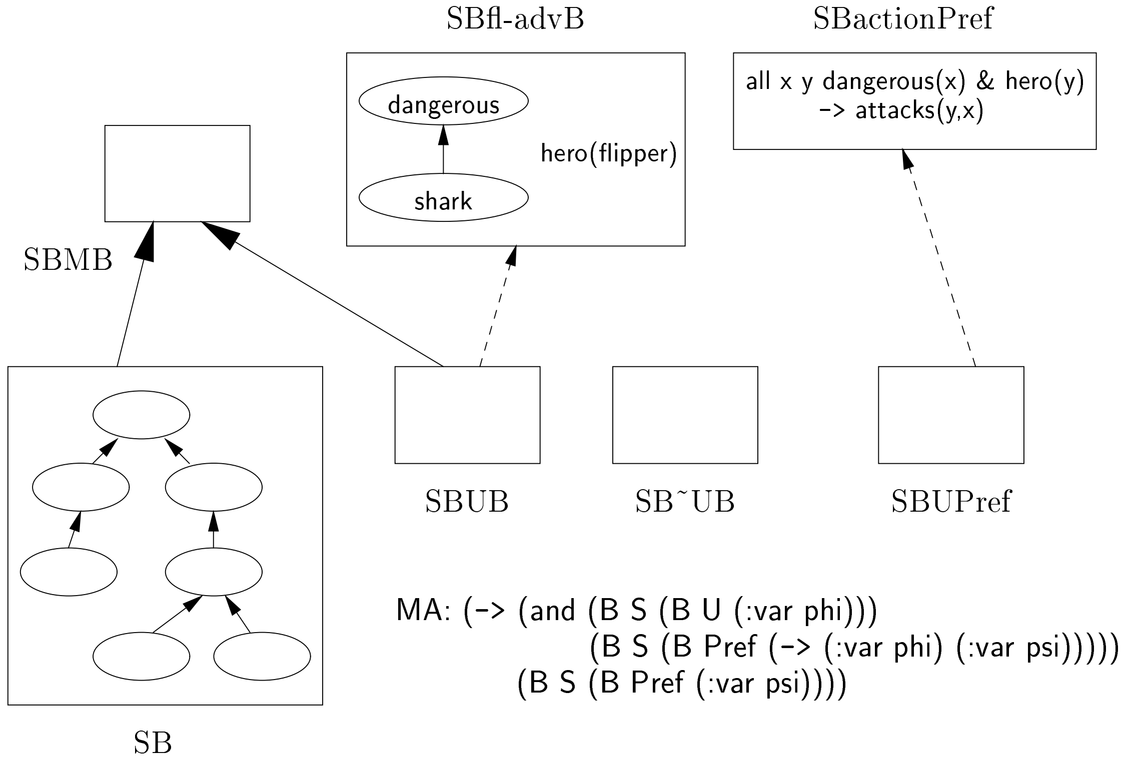


Figure 7.3: A BGP-MS UMKB after development time

The description will however focus on the inputs to BGP-MS, which were explicitly mentioned in the previous sections. FOPC translations or translation frames for AL^+ reasoning are specified.

As any AsTRa UMKB, the BGP-MS UMKB that results from the development session sketched before is of the form $\langle \mathcal{AT}, \mathcal{S}, \mathcal{MF}, \mathcal{MA} \rangle$. The four UMKB parts are as listed in Table 7.3. Figure 7.3 gives a more graphic overview of this UMKB. The assumption type partitions and stereotype partitions are shown with their most relevant contents (the SB concept hierarchy is only sketched—see Figure 7.2 for details). In the lower right corner, the modal axiom that relates SBUB assumptions to SBUPref assumptions is displayed.

7.2 Run Time

This section illustrates how BGP-MS can be used by a running user modeling application. BGP-MS explicitly distinguishes two performance modes for development time and run time. For instance, stereotype conditions are evaluated in run time mode only. BGP-MS allows the developer of a user modeling system to switch into run time mode during the development phase already, e.g. for testing purposes. Then, function calls will typically be used. When BGP-MS is initialized for use with running applications, it will automatically enter run time mode. Then, applications will communicate with BGP-MS using inter-process messages. In the following, we assume that our example application is running and communicating with BGP-MS. Hence, we will present BGP-MS messages[3].

7.2.1 Adding User Model Contents

In the following, we will assume that both stereotypes, 'action' and 'fl-adv', have been activated, so that the contents of stereotype partitions SBactionPref and SBfl-advB are inherited by the view knowledge bases of SBUPref and SBUB, respectively.

The following example illustrates the use of both negative type reasoning in case of a consistency check and forward reasoning for user model acquisition. The reasoning levels for consistency check and forward reasoning within **bgp-ms-tell** will be determined by setting parameter options.

Imagine a "Flipper" scene, where an actor in a boat points to an object in the water and utters the sentence:

$$\text{Watch out, a shark!} \tag{7.2}$$

Afterwards, the spectator is assumed to believe that the object, which is internally denoted by shark56, is a shark. To add a corresponding user model content into the UMKB, the message

```
(bgp-ms-tell (SBUB "shark(shark56)")
             :consistency NA :forward TI)
```

can be sent.

:consistency NA makes BGP-MS use reasoning level NA, i.e., specialized negative assumption type reasoning for checking consistency. Hence, this option leads to the application of the function $\mathbf{consistent}_{NA}$ and hence of the negative type reasoning function *na-consistent* (cf. Section 5.4). The latter function, according to the set inconsistent(SBUB), looks in SB∼UB for –shark(shark56) by calling

$\mathbf{derivable}_{TI}(\mathsf{SB{\sim}UB{:}{-}shark(shark56)})$

Since SB∼UB is a negative type, $\mathbf{derivable}_{TI}$ will not invoke type-internal reasoning. Therefore, and since –shark(shark56) is an FOPC expression, the above call reduces to

$\mathit{fetch}_{FOPC}(\mathrm{KB}_{SB\sim UB}, \mathsf{-shark(shark56)})$

[3]Note that in messages, parameters need not be quoted with the quote character '.

Since $\mathrm{KB}_{SB\sim UB}$ is empty, $fetch_{FOPC}$ fails. Therefore, shark(shark56) is consistent with the current UMKB according to reasoning level NA (reasoning at other levels would have brought the same result).

:forward TI makes BGP-MS use the TI reasoning level for forward reasoning, i.e., the type-internal forward reasoning function $\mathbf{forward}_{TI}$ is applied in addition to **store**. Since the assumption content $ac =$ shark(shark56) is an expression of $\mathcal{L}_{FOPC}$, FOPC is the applicable content formalism $F(ac)$. That is why BGP-MS calls

$$forward_{FOPC}(\mathrm{KB}_{SBUB}, \mathsf{shark(shark56)})$$

The $forward_{FOPC}$ function, on its part, employs KN-OTTER for making the forward inferences. Thus, $forward_{FOPC}$ finally invokes (cf. Section 6.3.2)

$$kn\text{-}otter^*(\mathrm{KB}_{SBUB}, \{\mathsf{shark(shark56)}\})$$

*kn-otter**, i.e. the version of *kn-otter* that returns derived resolvents, starts a KN-OTTER process and passes it the FOPC contents of KB_{SBUB}. In addition, KN-OTTER will also take the SB-ONE contents of the currently considered view into account. Due to inheritance from the 'fl-adv' stereotype, respectively from its stereotype partition SBfl-advB, the view knowledge base of SBUB contains the IS-A relation (isa shark dangerous). In this specific case, KN-OTTER picks this relationship from SB-ONE, because the subconcept shark corresponds to the predicate of the literal shark(shark56), which KN-OTTER attempts to resolve with contents of the knowledge base. The IS-A relation is handled like its logical translation

$$\forall x\, \mathrm{shark}(x) \rightarrow \mathrm{dangerous}(x)$$

so that one of the resolvents that *kn-otter** returns as result of the forward reasoning process is:

dangerous(shark56)

For this $\mathcal{L}_{FOPC}$ expression, the translation frame is computed:

("dangerous(" ",shark56)")

Then, the pair

$$\langle\ \mathsf{dangerous(shark56)},\ (\texttt{"dangerous(" ",shark56)"})\ \rangle$$

is entered into KB_{SBUB} and marked as an implicit user model content (cf. next section and Section 7.3.2).

In the given scenario, a possible use of forward reasoning could be the simulation of hypothetical user inferences in order to determine the appropriate content of system output (i.e., utterances of the "Flipper" actors in this case). Such an approach has been pursued in content planning (e.g., by [Zukerman and McConachy, 1993]; cf. Section 4.1.6) in order to avoid details which the user can infer on her own. In our example, the above bgp-ms-tell message could have been sent before the actor utterance concerning the object shark56. The application would wait for the forward reasoning process to finish and then query BGP-MS about facts that are related to possible alternatives for the next utterance. For instance, if forward reasoning had not derived that the user can be assumed to know that the object shark56 is dangerous, the system could have made an actor say

What a dangerous beast!

in addition to utterance (7.2). However, since the current user could be inferred to know the fact to be conveyed with this utterance, it can remain unsaid.

The next section describes how the above-mentioned queries can be made in BGP-MS.

7.2.2 Querying the User Model

Consider again the situation with the man in the boat who sees a shark. A rule of the content planner could be that concepts, which are explicitly assumed to be unknown to the user, should not be uttered without further explanation. A similar heuristic is used, e.g., in the adaptive hypertext system KN-AHS [Kobsa *et al.*, 1994]. KN-AHS adds explanatory information to hypertext pages concerning concepts of the text domain that the user does certainly not know according to the user model. So, before utterance (7.2), the application could have queried the user model about the user's beliefs about the concept shark. In case the concept had not been known to the user, an alternative to (7.2) might have been

Look at the beast over there, with the triangle fin!
That's a shark.

For queries, BGP-MS provides the **bgp-ms-ask** function. Similar to **bgp-ms-tell**, its main parameter is a user model content expression. However, in case of queries, the range of possible expressions is more limited than in case of entries to the UMKB. For **bgp-ms-ask**, BGP-MS allows view-internal expressions only, which can be formulated as $V{:}ac$ pairs or as simple modal formulas within the limits of $AL^{\neg}$. This means that queries concerning positive and negative assumptions are possible. In principle, the modal reasoning facilities would also permit full modal formulas as query expressions. However, complex modal formulas are not likely to be assumptions about the user, which an application is interested in. Furthermore, modal reasoning will be much more focussed and therefore more efficient, if the goal to be proven does not exceed $AL^{\neg}$.

The following query could have been passed to BGP-MS in the example above:

```
(bgp-ms-ask (SB~UB (:concept shark))
            :derivable NA)
```

When the bgp-ms-ask message (and hence the **bgp-ms-ask** function) is used without any parameter option, then it will not employ backward reasoning to answer the query. However, in this example, negative assumption type reasoning shall be employed. This is the only sensible reasoning level for this query. SB∼UB is a negative type, so that basic type-internal reasoning at levels TI or $TI + NA$ is not applicable. Moreover, type-external reasoning cannot be useful here, because the only type-external expression in the UMKB, the modal axiom, is not concerned with SB∼UB assumptions.

First, **bgp-ms-ask** invokes the **fetch** access function to check if the queried expression is explicitly contained in the UMKB. Since (concept shark) is an expression of $\mathcal{L}_{SB\text{-}ONE}$, BGP-MS uses SB-ONE mechanisms for handling the query. SB-ONE is a simple content formalism, so that a call to $fetch_{SB\text{-}ONE}$ is transformed into a call to $ask_{SB\text{-}ONE}$. Hence, BGP-MS invokes

$ask_{SB\text{-}ONE}(\mathrm{KB}_{SB\sim UB},$ (concept shark))

This call returns **false**, since the concept shark is not contained in view SB∼UB. Then, **bgp-ms-ask** attempts to derive the query expression using the facilities of the specified reasoning level NA. According to the set derivable(SB∼UB), the **derivable**$_{NA}$ function would have to look for the negation of the queried assumption content in view SBUB. However, since SB-ONE is a formalism without negation operator, this is obsolete. After all, the query is answered negatively, i.e., BGP-MS sends the following reply message to the application:

```
(bgp-ms-answer :answer no)
```

Now, consider the forward reasoning example of the previous subsection again. BGP-MS was able to derive SBUB:dangerous(shark56) from a new entry to the UMKB. The application now needs to know if the user is assumed to believe in this fact. It should transmit the query

```
(bgp-ms-ask (SBUB "dangerous(shark56)") :derivable TI)
```

Note that, as an alternative to the view-internal expression,the application could have employed the equivalent $AL^{\neg}$ expression, thus accessing the UMKB in a top-down fashion:

```
(bgp-ms-ask (B S (B U (dangerous shark56)))
            :derivable TI)
```

Since the $AL^{\neg}$ expression (B S (B U (dangerous shark56))) can be identified as corresponding to the view-internal expression SBUB:dangerous(shark56), both queries are handled identically. The :derivable TI parameter option determines that BGP-MS will consider type-internal implicit user model contents for answering the query. However, again the **fetch** access function is invoked first to check if the query expression is an explicit UMKB content. Since an FOPC content expression is concerned,

$$fetch_{FOPC}(\mathrm{KB}_{SBUB}, \mathsf{dangerous(shark56)})$$

is invoked. Recall that dangerous(shark56) has been entered into view SBUB, after it was inferred to be an SBUB assumption by type-internal forward reasoning. However, it was marked as an implicit UMKB content, since it originates from an inference process. Hence, **fetch** is not successful. Then, the function $\mathbf{derivable}_{TI}$ is called, which invokes $derivable_{FOPC}$ in this case. Like all backward reasoning functions in BGP-MS, it is required to first look for implicit entries in the current view before starting a backward reasoning process to answer a query. Thus, queries using backward reasoning may benefit from previous forward reasoning efforts. In our example, dangerous(shark56) is found to be stored in view SBUB as an implicit UMKB content, so that the query can be answered positively:

```
(bgp-ms-answer :answer yes)
```

The procedure of handling forward-derived UMKB contents may seem somewhat complicated, but there is a philosophy behind it. Implicit user model contents shall be handled the same, no matter whether they were derived by forward or by backward reasoning. If a $fetch_F$ function considered the results of previous forward reasoning processes, which are contained as implicit entries in a view knowledge base, the status of forward and backward reasoning results would be different, since $fetch_F$ at the same time would not consider backward-derivable implicit contents.

Back to our example application: Now that the shark has appeared and that the user has been informed about this, the system wants to know about the preferences of the user concerning the next scene. Imagine that there are several options as to what will happen next. So, the system can send queries to BGP-MS concerning these options, among them

```
(bgp-ms-ask (B S (Pref U (attacks flipper shark56)))
            :derivable AL+)
```

The :derivable AL+ parameter option makes BGP-MS use view-external AL^+ reasoning for answering this query. The use of this option makes perfect sense in this case, since the developer defined an AL^+ axiom that generally permits assumptions about user preferences to be inferred with the help of assumptions about user beliefs. If the AL^+ level is chosen for derivation, then all other kinds of reasoning, namely view-internal reasoning and negative type reasoning, are superfluous; view-external reasoning subsumes both methods.

For answering queries at the AL^+ level, the function $\mathbf{derivable}_{AL^+}$ is utilized, which on its part invokes *e-derivable*, the specialized AL^+ reasoning function of BGP-MS (cf. Section 6.4.2). So, the above query results in the call

$$e\text{-}derivable(\text{UMKB}, \Box_{(B,S)}\Box_{(Pref,U)}\text{attacks}(\text{flipper}, \text{shark56}))$$

Finally, KN-OTTER is employed for doing the inference work. For checking AL^+ derivability, the *kn-otter* function needs the functional translation of the UMKB, T(UMKB), and the translation of the negated query formula (cf. p.169). In Section 6.4.2, an optimized method for on-demand computing of T(UMKB) was presented. The translations of modal formulas and modal axioms are available in the UMKB, but the translations of view-internal contents still must be composed. This composition of view-internal translations is illustrated using a simple example. One of the UMKB contents that are necessary to derive the query is SBUB:dangerous(shark56). In order to compose the translation of this V:ac expression, three pieces of data are needed, which are already available in the UMKB (cf. Section 7.2.1 and Section 7.1.2):

1. the translation frame dangerous(shark56)[ ] = (`"dangerous(" ",shark56)"`)
2. the quantifier list $Q(SBUB)$ = `"all f1 all f2"`
3. the world term $wt(SBUB)$ = `"apply(f2,B,U,apply(f1,B,S,w0))"`

The translation of a view-internal expression V:ac is composed according to definition 6.4 (i.e., $T(V:ac) := Q(V)\, ac[wt(V)]$). So, we get

T(SBUB:dangerous(shark56)) =
`"all f1 all f2 dangerous(apply(f2,B,U,apply(f1,B,S,w0)),shark56)"`

For the derivation of (B S (Pref U (attacks flipper shark56))), the translations are needed of

- some view-internal user model contents, namely SBUB:dangerous(shark56), SBUB:hero(flipper), and the preference rule SBUPref: all x y dangerous(x) & hero(y) -> attacks(y,x). Their translation is generated by T using on-demand composition, like illustrated by the example above.
- the modal axiom. Its translation is already available in the UMKB.
- the negated query (not (B S (Pref U (attacks flipper shark56)))). Since it does not correspond to a view-internal expression (assumption types must not start with ~SB), it will be translated using the functional translation function Π_F.

With these inputs,

$$kn\text{-}otter(T(\text{UMKB}), \Pi_F(\Box_{(B,S)}\Box_{(Pref,U)}\text{attacks}(\text{flipper}, \text{shark56}))$$

returns successfully; the query is derivable from the UMKB via view-external reasoning.

In the previous example, AL^+-level backward reasoning was used to answer an application query concerning a user preference. The query was issued to determine whether the current user would prefer one from a set of possible subsequent actions in the Flipper show or not. Similar queries would have to be submitted for every possible continuation. If the set of continuations is large, this method will become very expensive, particularly if preference options deal with novel events or protagonists. In this case, the user model will certainly not explicitly contain assumptions about user-related preferences, so that backward reasoning needs to be done.

In our example, it would probably have been more efficient to let BGP-MS determine user preferences beforehand with the help of forward reasoning (on the AL^+ level), which in principle can be performed as background process. This could have been accomplished by using a :forward AL+ parameter option instead of :forward TI in the call to **bgp-ms-tell** in Section 7.2.1:

```
(bgp-ms-tell (SBUB "shark(shark56)")
             :consistency NA :forward AL+)
```

Thence, $\mathbf{forward}_{AL^+}$ (and, consequently, *e-forward*) would have been used instead of $\mathbf{forward}_{TI}$ (and, consequently, $forward_{FOPC}$). From the view-internal input, other view-internal knowledge, and the modal axiom, AL^+ forward reasoning could have inferred the modal formula

$$\Box_{(B,S)}\Box_{(Pref,U)}\,\mathrm{attacks}(\mathrm{flipper},\mathrm{shark56})$$

This AL formula corresponds to the view-internal expression

SBUPref:attacks(flipper,shark56)

so that the new assumption content attacks(flipper,shark56) would have been entered into view SBUPref as an implicit content. In this case, the above query could have been answered without reasoning, because the $\mathbf{derivable}_{AL^+}$ function would have found the query expression to be stored as an implicit UMKB content.

7.3 Related Components

This section presents, for sake of completeness, other components of BGP-MS that are related to the central representation and reasoning mechanisms. Besides forward and backward reasoning for acquiring secondary assumptions, there are two mechanisms that can support the acquisition of primary assumptions about the user. These mechanisms, namely *dialog act types* and *user interviews*, will be dealt with first.

Thereafter, we describe the maintenance of so-called *source information*, which BGP-MS assigns to assumption contents in order to store information about how an assumption was acquired. Possible sources of UMKB contents are the primary and secondary inference components of BGP-MS, in addition to direct observation by the application.

BGP-MS user modeling knowledge bases are structured by means of assumption types. However, there is no "thematical" structure in the UMKB. Particularly in a centralized user modeling scenario, where the user modeling system interacts with several applications, it may be possible to identify modules of a UMKB—related to distinct aspects concerning the user or an application—which might be interesting to more than one application. For instance, in a suite of office software, there may be general assumptions about the user's proficiency in using the suite as a whole, but also specialized user model contents concerning single suite components only, like spreadsheets, email, etc. BGP-MS offers *domains* to divide UMKBs into modules. As already stated, domains might be especially useful in centralized user modeling scenarios. So, on the basis of domains, BGP-MS was made a user modeling server which can handle several applications as well as several users at a time. The idea of *domain-based user modeling* will be sketched briefly in Section 7.3.3.

A prerequisite for becoming a server is the capability to exchange inter-process messages with applications. Quite early, BGP-MS was enabled to do inter-process communication, but only with applications running on the same computer. In order to allow network-wide communication and for reasons of standardization, BGP-MS now employs the agent communication language KQML for its *inter-process communication interface*, which will described at the end of this section.

7.3.1 User Model Acquisition

Dialog Act Types

Heuristic rules have frequently been employed in the user modeling literature for acquiring assumptions about the user. Particularly if these rules serve for primary acquisition, they are often domain-specific, but there are also a number of domain-independent heuristics, which normally describe the assumptions that can be made when the user performs specific actions at the user interface. The assumptions that are made with many of these heuristics can be regarded as prerequisites to the successful execution of the respective actions. For example, the correct use of an object presumes that the user knows the object; hence she can be assumed to know the object if she uses it correctly [Chin, 1989; Nwana, 1991; Sukaviriya and Foley, 1993]. A request for additional technical details about an object is likely to presume that the user is familiar with this object; hence she can be assumed to know the object if she requests technical details for it [Boyle and Encarnacion, 1994].

In BGP-MS, *dialog act types* were introduced that permit the definition of such straightforward, action-oriented rules for primary acquisition [Pohl *et al.*, 1995]. Then, a *dialog act* is an action of the user that is an instance of one of the available dialog act types. A dialog act type determines what can be inferred from instantiating dialog acts, and under what preconditions the inference can be made. The naming of dialog acts and dialog act types is due to their relationship to speech acts [Searle, 1969] and to the presupposition analysis method for the acquisition of dialog partner models in natural-language dialog systems [Kobsa, 1985]. This method analyzes a user utterance with respect to the speech acts it verbalizes, and from each speech act derives the so-called *presuppositions*, which must have been valid for the speaker to perform the acts correctly. This is especially interesting if the derivations can be made without regard to the content of the speech act, i.e. if they are only determined by its type (like a 'question' or an 'inform' act).

In order to enable the user model developer to define acquisition rules that are independent of action content, dialog act types can be parameterized and associated with a set of *presupposition patterns* that schematically describe the presuppositions of all instances of the dialog act type. After a set of such types along with their presupposition patterns has been defined and collected in a library of dialog act types, *dialog act analysis* can be performed: the presuppositions of an observed dialog act can be computed by suitably instantiating the presupposition patterns of its type.

We found that there are only few dialog act types that seem to be generally applicable in all kinds of interactive computer systems. However, the situation improves if more specific "application scenarios" are considered. BGP-MS therefore offers the developer not only a library with general dialog act types, but also a library with types that are of general use in the area of adaptive hypertext systems. Hypertext dialog act types were investigated in connection with the development of the adaptive hypertext system KN-AHS [Kobsa *et al.*, 1994]; the whole primary acquisition of KN-AHS can be done with dialog act types from the hypertext library. In addition, the developer may define new dialog act types that may serve as application-specific acquisition rules.

A simple example is a dialog act type from the hypertext library that can be used if the system provides hypertext links from text elements to explanatory information. It is defined with the define-d-act Macro of BGP-MS:

```
(define-d-act :name REQUEST-EXPLANATION
              :parameters (topic)
              :presuppositions ((B S (not (B U topic)))))
```

The REQUEST-EXPLANATION type has one parameter, which is the topic to be explained by the application. In the presupposition list (:presuppositions parameter), patterns of view-internal or $AL^{\neg}$ expressions are allowed. In this case, there is only one presupposition pattern in form of an $AL^{\neg}$ expression. It prescribes that, if an instance of REQUEST-EXPLANATION occurs, it will henceforth be assumed that the user does not know the concerned topic. The inference is carried out by applying **bgp-ms-tell** to the elements of the presupposition list, the pattern variables of which were instantiated with the parameter of a specific dialog act. In addition to a presupposition list, a dialog act type definition may also contain a list of preconditions. If preconditions are present, then the presuppositions can only be inferred if all preconditions are satisfied. Also preconditions are view-internal or $AL^{\neg}$ expressions; **bgp-ms-ask** is employed to evaluate them. So, dialog act inferences can depend on the current user model. To this respect, they are similar to the inference rules of [Zukerman and McConachy, 1993], which infer user beliefs from system utterances, possibly taking the user's current beliefs into account (cf. Section 4.1.6).

The occurrence of a dialog act is reported to BGP-MS with a d-act message, which contains the name of a dialog act type and a list of dialog act parameters. An example, which might occur in a hypertext on computer operating systems that offers an explanation node for the concept "kernel", is

```
(d-act REQUEST-EXPLANATION ((:concept kernel)))
```

This dialog act results in

```
(bgp-ms-tell '(B S (not (B U (:concept kernel)))))
```

User Interviews

A second possibility of doing user model acquisition with BGP-MS is to make use of the interview component. Initial interviews are a major source of information about the user [Rich, 1979; Peter and Rösner, 1994; Boyle and Encarnacion, 1994]. BGP-MS lets the developer declaratively define user interviews. Based on an interview definition, BGP-MS will control the execution of an interview, when the application invokes it. I.e., it determines the questions to be asked, it draws inferences about the user based on her answers, and enters the derived assumptions into the user model. The display of interview questions and the receipt of answers can be performed by the application, but BGP-MS also offers a component which can perform these tasks.

A BGP-MS interview consists of a number of question blocks, each of which can consist of one or more questions. The questions of a question block are presented to the user all at once, and the user's answers to the questions of one block are sent back to BGP-MS as a whole. At any time, the user may choose to skip individual questions, question blocks, or even the whole interview. In an interview definition, a sequence of question blocks is fixed, and a question block is defined as a list of questions.

The definition of a question is more complex. It includes a name, the question text, the question type, and the conclusion part. The question type specifies the set of possible answers, which must be taken into account when the question is visualized. Possible question types include yes-no-questions, questions with a predefined set of answers (out of which one or more can be selected), and questions with a numeric range of answers. Finally, in the conclusion part, the developer can define the assumptions that should be derived from specific answers using view-internal expressions or $AL^{\neg}$ expressions.

```
(define-interview-question
  :name flipper-experience
  :text "How often do you watch Flipper?"
  :type (:select-one ("regularly" "sometimes" "never"))
  :conclusions (((equal answer "never")
                  (B S (fl-novice *current-user*)))
                 ((equal answer "regularly")
                  (B S (fl-adv *current-user*)))))
```

Figure 7.4: Definition of an interview question

Fig. 7.4 shows the definition of an interview question by which the user's degree of familiarity with the "Flipper" series is to be determined. The possible answers form a set containing the elements "regularly", "sometimes", and "never", only one of which may be selected. If the answer is "never", **bgp-ms-tell** is applied to the associated conclusion, so that the expression fl-novice(*current-user*) will be entered into partition SB. If the answer is "regularly", the expression fl-adv(*current-user*) will be entered into SB. Otherwise, no new entries will be made.

The conclusions about the user are drawn as soon as the answers to a question block are returned to BGP-MS with the help of an interview-response message. A possible answer to a question block that contains the above question is

```
(interview-response ... ( ... (:answer "regularly") ... ))
```

According to the question definition, this answer leads to the function call

```
(bgp-ms-tell '(B S (fl-adv *current-user*)))
```

This interview conclusion satisfies the activation condition of stereotype 'fl-adv', as defined in Section 7.1.2. It is not unusual that interview answers are used to determine initial stereotype activations.

7.3.2 Source Information

Entries into the current user model may have different sources: they can be observed by the application or can be inferred by either the interview component, the dialog act analysis component, stereotype activation, or inferences performed within the user model. For each entry, BGP-MS stores its source in form of a source symbol. In addition, BGP-MS has several other types of information

source symbol	:origin	:category	:reliability
:application	:observed	:primary	:high
:developer	:pre-defined	:primary	:high
:KN-DACT	:inferred	:primary	:medium
:interview	:inferred	:primary	:high
:KN-STEREO	:stereotype	:secondary	:medium
:reason	:inferred	:secondary	:medium

Table 7.4: Source symbols and associated properties of user model contents

associated to user model entries that depend on its source: a status information (inferred vs observed), the reliability of the information, etc. An application can make use of **bgp-ms-ask** to retrieve such information from BGP-MS.

As a user model entry may be based on more than one source (e.g., it may have been observed by the application and additionally inferred by internal inferences), there may also be multiple values attached to one information slot.

The following source symbols exist and are defined to be attached to a user model entry under the following circumstances:

:application The entry has been made as result of a call to **bgp-ms-tell** at run time.

:developer The entry is the result of a call to **bgp-ms-tell** at development time.

:interview The entry has been made as a result of evaluating the conclusion part of an interview question definition.

:KN-DACT The entry is a result of dialog act analysis performed because of an incoming d-act message.

:KN-STEREO The entry is contained in a stereotype partition.

:reason The entry has been found to be deducible from the current user model by forward or backward inferences at any reasoning level.

Currently, BGP-MS associates three information slots (:origin, :category, and :reliability) with each source symbol. Table 7.4 shows the values of these slots depending on the different source symbols.

An additional parameter option of **bgp-ms-ask** can be used to retrieve source information from the UMKB. In the Flipper example, it was inferred by forward reasoning that the user believes the object shark56 to be dangerous. A bgp-ms-ask message with a :derivable TI option could retrieve this implicit SBUB assumption from the UMKB. The extended message

```
(bgp-ms-ask (SBUB (dangerous shark56))
            :derivable TI :return (:origin :reliability))
```

would not simply lead to a positive reply, but would cause the application to be informed about the requested source information. In this case, the reply message is

```
(bgp-ms-answer :answer (:inferred :medium))
```

7.3.3 Domain-Based User Modeling

So far, we have been concerned with a single knowledge base that contains all knowledge that is needed in the user modeling process. At development time, a-priori user modeling knowledge that is related to an application is entered into such a user modeling knowledge base. Immediately after development time, a UMKB is still generic, since it contains no information about an individual user. At run time, assumptions about the user are added to the UMKB, thus individualizing its contents. Hence, for every user a separate instance of the UMKB in its generic, post-development state is needed. Then, at run time a UMKB is determined by the application which its a-priori contents are related to, and by the user whose individual user model is contained in the UMKB. However, there are user modeling scenarios where such application/user (short: (A, U)) knowledge bases are suboptimal. First, a division of the UMKB into modules is interesting, if the UMKB covers several user model aspects which are unrelated to each other. Second, if user modeling is done by a central component that serves the user modeling needs of several applications, it could be beneficial to applications to share user model contents, particularly if they are concerned with the same user model aspects. With an (A, U) UMKB this is not possible, since it is limited to one application only.

For BGP-MS to permit both modularization of UMKBs and sharing of UMKB contents, *domains* were introduced [Pohl and Höhle, 1997]. Taken precisely, it is not *application* data that is set up at development time, but rather application *domain* data. Therefore, the application/user UMKBs mentioned above should better be regarded as domain/user UMKBs (D, U) that store the assumptions about a user U concerning a domain D in addition to the development data of D. Now, there is no reason to constrain an application to one domain only. The user modeling data needed by an application A_i could be modularized into several domains $D_1, \ldots, D_n$. Vice versa, there is no reason to constrain a domain to be used by one application only. So, another application A_k could make use of one or more domains from A_i and other applications, in addition to defining its own domains. Now, if a domain D_y is shared by two applications A_i and A_k, two instances of these applications that are being used by user U_r will share the (D_y, U_r) UMKB, besides using other "private" UMKBs (cf. Figure 7.5).

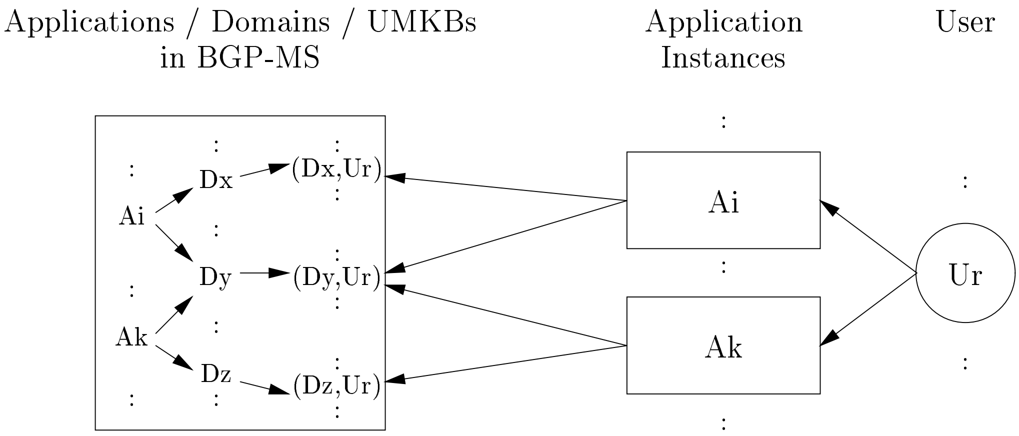


Figure 7.5: Applications, domains, and users

Thus, a new paradigm for organizing user modeling knowledge has been created, namely *domain-based user modeling*. The main advantage of this approach is that user modeling knowledge needed by an application may be structured into modules and that this knowledge and the corresponding user model contents can become shared by other applications.

For domain-based user modeling, several new functionalities for both development time and run time are needed. Most important for development time is that after setting up all domain user modeling knowledge, the domain is saved and made accessible to applications. According to the AsTRa specification, BGP-MS has always one current UMKB. So, both at development time and at run time it must be possible to switch between UMKBs. At run time this is done by specifying the concerned domain and the current user in the messages to BGP-MS. An example is

```
(bgp-ms-tell (SBUB "shark(shark56)")
             :user WP :domain flipper)
```

For processing this bgp-ms-tell message, BGP-MS will switch to the (flipper, WP) UMKB, i.e. the UMKB that stores the user model of user WP with respect to the flipper domain, along with flipper domain user modeling knowledge.

7.3.4 Inter-Process Communication with BGP-MS

At development time, the typical way of communicating with BGP-MS is to use its functional Lisp interface. At run time, however, the standard way of interaction between BGP-MS and an application is the exchange of inter-process messages. The types of messages that can be sent to BGP-MS correspond to the functions of the Lisp interface; mainly, there are bgp-ms-tell, bgp-ms-ask, d-act, and interview-response messages. In addition, bgp-ms-answer messages are sent by BGP-MS to the application and contain the answer to a previous query.

Until recently, inter-process message communication between BGP-MS and an adaptive application used a proprietary communication system for synchronized communication between processes on one computer. Particularly if BGP-MS is to be used in combination with several applications, i.e. as a kind of user modeling server [Orwant, 1995], a communication mechanism is needed that allows asynchronous communication with applications that can address BGP-MS from other computers across a network.

KQML (Knowledge Query and Manipulation Language) was developed as a communication language for knowledge-based agents [Finin *et al.*, 1994]). Typical KQML implementations satisfy the above-mentioned criteria, i.e. they offer network communication facilities as well as asynchronous communication procedures. KQML provides a set of performatives (i.e., message types) mainly for manipulating and accessing the knowledge bases of other agents, but also for organizing and controlling communication. BGP-MS can be regarded as a knowledge-based agent, its knowledge base being the current UMKB. Therefore, KQML appeared to be an appropriate candidate for improving the inter-process communication facilities of BGP-MS. Since KQML is a proposed standard that has emerged from the ARPA knowledge sharing effort [Patil *et al.*, 1992], it is known and commonly used in the agent world. If it is employed by BGP-MS, it can be expected that user modeling servers based on BGP-MS will be compatible with quite a range of existing systems, and that it will be easier for application developers to establish communication with a BGP-MS server. The usefulness of a communication facility that is based on KQML-like message types is also pointed out in [Paiva and Self, 1995], where the interface messages of the user and learner modeling shell TAGUS are related to KQML performatives.

Syntactically, KQML messages are Lisp-like structures that consist of a performative name and a list of parameters (following parameter keywords). There are some standard parameters that occur in most performatives. :sender and :receiver parameters must be used to specify sender and receiver of a message. A message sender can employ the :reply-with parameter to specify an identifier that the receiver should use as value of the :in-reply-to parameter of a response message. Thus, messages can become related to each other. The parameters :content and :language are used to specify the message content and the language it is formulated in. In BGP-MS, the exact use of these parameters varies slightly between the different performatives.

BGP-MS employs two kinds of KQML performatives (i.e., message types), namely content-oriented ones and administrative ones. The latter are mainly concerned with the maintenance of user and domain information, while the former correspond to the standard BGP-MS messages that were listed above. We will briefly present the most important content-oriented performatives here, which are related to the representation and reasoning facilities of BGP-MS:

```
(tell :sender tcp://diva:8094
      :receiver tcp://asterix:8091
      :language VI :content (SBUB "shark(shark56)")
      :user WP :domain flipper)

(ask-if :sender tcp://diva:8094
        :receiver tcp://asterix:8091
        :language VI :content (SBUB "dangerous(shark56)")
        :reply-with query23
        :derivable TI
        :return (:origin :reliability)
        :user WP :domain flipper)

(reply :sender tcp://asterix:8091
       :receiver tcp://diva:8094
       :in-reply-to query23
       :content (:answer (:inferred :medium)))
```

Table 7.5: KQML communication with BGP-MS

- The performative tell is the main means for transferring information to BGP-MS. It replaces the previous message types bgp-ms-tell, d-act, and interview-response. BGP-MS infers the information format from the :language parameter and processes the :content accordingly. Possible :language values are VI for view-internal expressions, AL- for $AL^{\neg}$ expressions, D-ACT for dialog acts, and IV-RESPONSE for interview answers.

- The query performative ask-if is handled like the former bgp-ms-ask message type. VI and AL- are the only permitted values of the :language parameter, and the queried expression becomes the message content.

- The performative reply is used to send response messages. That is, reply replaces the former bgp-ms-answer message.

Since KQML was not designed for user modeling, there was a need for extensions, which are tolerated by the KQML specification [Finin *et al.*, 1993], and for modifications. First, the performatives tell and ask-if were extended to carry the parameter options of **bgp-ms-tell** and **bgp-ms-ask**, which control the reasoning of BGP-MS. Second, in order to make them suitable for domain-based user modeling, tell and ask-if were further extended with :user and :domain parameters. They serve to identify the affected domain/user UMKB. Third, administrative performatives for handling domain and user information were added to the set of KQML performatives.

Table 7.5 shows an example exchange of KQML messages between BGP-MS and an application. Again, we make use of the "Flipper" scenario and show how some of the UMKB inputs and queries of Section 7.2 can be communicated as KQML messages. First, there is a tell message which enters the view-internal expression SBUB:shark(shark56) into the UMKB. Second, the UMKB is queried with ask-if about the dangerousness of shark56. Third, the reply message that BGP-MS sends is shown. In the ask-if and reply messages the use of message identifiers (see :reply-with and :in-reply-to) and of source information is illustrated. The :sender and :receiver values are TCP URLs that identify communication ports on the specified computers. The simple computer addresses (diva and asterix) indicate that communication takes place on a local network; BGP-MS runs on asterix using port 8091, and the application runs on diva using port 8094.

Part III

Discussion

Chapter 8

Related Systems

In this chapter, several systems will be discussed and compared to BGP-MS and the AsTRa framework. First, a general, multi-purpose representation system is described, which can be used for belief modeling and is very close to the ideas of the AsTRa framework. Afterwards, the most important user modeling shell or tool systems that have been described in the literature will be reviewed. This review focuses on properties that are relevant to this thesis. For each system, it will be described

- how the UMKB is structured,
- whether and how different assumption types can be represented,
- what kind of reasoning the system is able to perform, and
- what interface to its representation system is provided.

Other overviews about user modeling shell systems are given for instance in [Kay, 1993] or in [Kobsa and Pohl, 1995].

8.1 RHET/SHOCKER

Among all knowledge representation systems, SHOCKER [Allen and Miller, 1993] is perhaps closest to the AsTRa framework. It was developed at the University of Rochester and is the successor of the perhaps more famous system RHET (RHETorical Knowledge Representation) [Allen and Miller, 1991; Miller, 1991]. Its main application objectives are natural-language understanding and planning; among other knowledge representation facilities, it provides a subsystem for temporal reasoning. In the following brief system review, the main representation mechanisms, the corresponding reasoning capabilities, and the interface for using SHOCKER will be discussed.

For knowledge representation, SHOCKER offers two basic formalisms. First, there is a powerful subsystem for frame-based representation. Second, a Horn clause theorem prover is available for propositional and rule-like knowledge. Horn clause inferences constitute the core of the reasoning capabilities of SHOCKER. Frame-based knowledge can be accessed by clauses, thus becoming integrated into Horn clause reasoning processes. This is similar to the integration of conceptual SB-ONE knowledge into type-internal FOPC reasoning in BGP-MS (cf. Section 6.3.2). In the following, we will only discuss Horn clauses in order to concentrate on the most relevant aspects of SHOCKER. An example clause is

[[above ?x ?y] < [on ?x ?z] [above ?z ?y]]

with the clause head (the conclusion) on the left and the clause body (the premises) on the right of the implication symbol <. As already indicated, such a clause can be regarded as a rule. The above clause is to express that an object is above a second one, if it is on a third one which itself is above the second one.

What relates SHOCKER closely to the AsTRa framework is that a knowledge base can be organized into different contexts. Contexts can be arranged in a context hierarchy with (multiple) inheritance. On the one hand, contexts support hypothetical reasoning. I.e., contexts can be used for maintaining competing hypotheses like those of a plan recognition system that is trying to interpret an action performed by some other agent. On the other hand, contexts may also be employed as “belief spaces” for the representation of beliefs of different agents. Typical belief spaces are labeled similar to AsTRa assumption types, with the exception that a leading ‘SB’ is not used for belief spaces concerning other agents. Examples of belief space labels are SB for system beliefs, AB for system beliefs about an agent A’s beliefs, and ABMB for what A believes to be shared (mutual) beliefs of S and A. Contexts can be used in both ways at the same time. In this case, there may exist alternative hypothesis contexts for a belief space.

At the user interface of SHOCKER, only hypothesis contexts can be addressed explicitly, while belief space information is expressed within clauses. A simple example for such clauses is [SB [p]], saying that the unit clause [p] is an element of belief space SB. But also more complex expressions are possible, like

[AB [[p] < [SB [p]]]]

This example demonstrates that belief spaces can be related to each other, like it is possible with type-external formulas in the extended AsTRa framework. Figure 8.1 gives an example of how the user view of hypothesis contexts with belief clauses is realized within SHOCKER as hierarchy of belief spaces which may have their own hypothesis contexts. In the user view on the left, there is a ROOT hypothesis context, which contains the above complex expression, and an inheriting hypothesis context C1. Note that arrows go downward in contrast to the upward arrows that are used for illustrating type/partition hierarchies of AsTRa/BGP-MS. These downward arrows denote that there is (downward) inheritance only. Dashed arrows represent context relations, while plain arrows represent belief

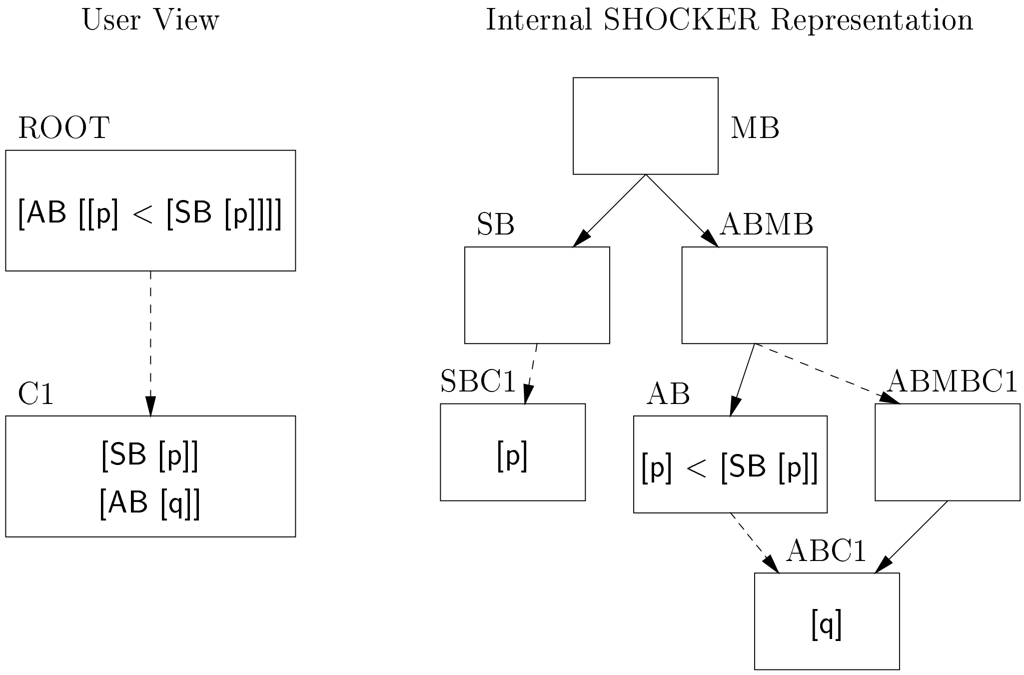


Figure 8.1: Belief representation in SHOCKER

space relations. Belief space relations are employed in the internal representation of SHOCKER. There, a standard hierarchy of belief spaces is constructed. In addition, copies of belief spaces are established (with labels '. . . C1') that altogether represent the second hypothesis context C1. A belief space may contain expressions that refer to other belief spaces, like the expression in AB demonstrates. It internally represents the complex expression which in the user view is contained in the ROOT context.

Since the internal representation of SHOCKER is partly hidden to its user, it can be said that SHOCKER permits top-down access to its facilities. In AsTRa systems like BGP-MS, both top-down and bottom-up access to representation facilities are possible.

The standard reasoning strategy of SHOCKER is Prolog-like backward chaining, which is used in answering knowledge base queries. In addition, forward chaining is available, but can only make use of clauses that contain one or more forward triggers. Within clauses, triggers appear after the premises, separated by a :forward keyword. Triggers are applicability conditions, which are evaluated after new data has been entered into the knowledge base. If a clause is applicable according to its triggers, it is processed in a normal forward chaining manner: the conclusion is entered into the knowledge base if the premises can be satisfied. Typically, triggers form a subsets of the set of premises, like in

[[above ?x ?y] < [on ?x ?z] [above ?z ?y] :forward [above ?z ?y]]

SHOCKER provides typical functions for adding knowledge and querying the knowledge base, e.g. 'assert-fact' and 'prove', which are similar to the **store** and **derivable** access functions of AsTRa. The function 'assert-if-consistent' only inserts a fact if it is consistent with the current knowledge base. The same strategy is pursued by the AsTRa **tell** function (cf. Section 5.2.4). In addition, knowledge base contents can be indexed in SHOCKER. Indices have no effect on reasoning, but can be used in database-like retrieval and removal functions.

In sum, SHOCKER is quite close to an AsTRa implementation, mainly because of its belief spaces that are a kind of assumption type knowledge bases. In particular, SHOCKER is similar to BGP-MS in using belief space hierarchies and a hybrid approach to content representation based on Horn clauses and frames. However, in contrast to AsTRa and BGP-MS, it pursues a top-down approach of integrating different mechanisms, with Horn clauses as the central formalism. By clauses, simple relationships between different belief spaces can be expressed. Both backward and forward reasoning are available, but besides functions for data-base-like access to the knowledge base, there is only one reasoning level.

There are two main shortcomings of the system: First, the set of content formalisms is fixed. Therefore, the limitations of Horn clause representation cannot be compensated by adding a more expressive formalism for belief space contents. Second, belief formulas are quite restricted in comparison to what is possible with modal formulas in the extended AsTRa framework. Neither negative assumptions nor general (i.e., schematic) relationships between different assumption types can be expressed.

8.2 GUMS

After UMFE [Sleeman, 1985], which was designed as user modeling front-end for tailoring expert system output to the user's conceptual knowledge, GUMS [Finin, 1989] is the first user modeling shell system. While UMFE could support a system in performing a specific adaptive task, GUMS was developed to be independent not only from application domains, but also from the tasks an application would try to accomplish.

GUMS is able to maintain user models for several independent applications. For each application, a tree-like stereotype hierarchy and a number of user models is stored. Individual models are attached as leaves to the stereotype hierarchy. Thus, the main information about a user is her classification as member of one of the predefined user groups. In addition, an individual user model may contain a collection of facts (i.e., positive or negative ground literals) about the user. At the time of interaction between an application and GUMS, a "current user" is determined. So, the UMKB for an active application consists of the stereotype hierarchy and the current user model.

Within stereotypes, facts and rules are formulated in a Prolog-like language, but with explicit negation of facts and literals. Both facts and rules can be declared to be either “definite” (i.e. necessarily true) or “default”. In addition, stereotypes may contain meta-level knowledge that mainly controls how negation is handled by the reasoning mechanism of GUMS. Within individual models, only definite facts are allowed. The use of default facts and rules leads to a four-valued logic. Possible truth values are ‘true’, ‘false’ (both based on definite facts and rules), ‘assumed(true)’, and ‘assumed(false)’ (when defaults are involved). Thus, default reasoning is used to express uncertainty.

The main reasoning process within GUMS is backward reasoning, which is performed to answer application queries. If more than one truth value can be determined for the queried literal, GUMS is optimistic and chooses the most positive truth value. Of course, if an expression can be derived to be both definitely true and definitely false, there will be an inconsistency. It can be resolved by reclassifying the user in the stereotype hierarchy.

There is no explicit level of distinction between assumption types in GUMS, neither within stereotypes nor within individual models. All assumptions about a user and all stereotyped assumptions concerning a user group are stored in coherent knowledge bases. However, specific predicates can be employed to express assumption type information, like in several user modeling systems (cf. Section 3.2.4). In this case, content information can only be expressed with logical terms. In the examples provided in [Finin, 1989], the predicate ‘knows’ is used to express SBUB assumptions, while the meaning of other expressions is less clear. The rules that are contained in stereotypes do not express implications which the user is assumed to know and to apply in her own reasoning. Such stereotype rules are meta-level inference rules that the system can apply in deducing assumptions about users that are classified to belong to the containing stereotype. Stereotyped assumptions about the user, however, are restricted to what is allowed in the individual user model, i.e., to facts.

As its interface to applications, GUMS provides six different commands. Four of them are of an administrative nature; they can be used to create a new user model, load an existing one, store the current user model, and finish a GUMS session. The other two commands are ‘add’ and ‘show’, which constitute a tell-and-ask interface to the current UMKB. In accordance with what can be assumptions about the user, arguments of both commands can be positive or negative literals.

When the application ‘add’s a fact, GUMS checks user model consistency by trying to deduce the negation of the fact. If an inconsistency is discovered, GUMS tries to assign the user to less specific stereotypes, which are higher up in the hierarchy, in order to resolve the inconsistency. So, GUMS tries to revise the user model in favor of new observations about the user. According to Finin, this strategy corresponds to the intuition that “the more we learn about an individual, the less we depend on stereotype reasoning” [Finin, 1989, p.422] by shifting to less informative stereotypes when reclassifying the user. However, the fact that by climbing up the stereotype hierarchy this revision process can only lead to a loss of stereotyped information may also be regarded as a shortcoming.

When GUMS is requested to 'show' a fact, it uses its backward deduction mechanism to determine one of the four possible truth values for the queried fact. Because negation-as-failure can be applied as a default rule, a query can at least be stated to be 'assumed(true)', when nothing is explicitly known about the fact. A query fact may contain variables; in this case, 'show' returns truth values for possible instantiations, due to backtracking.

In sum, the 'add' and 'show' interface of GUMS is in line with the definition of the **tell** (except forward reasoning) and **ask** access functions of the AsTRa framework (cf. Section 5.2.4).

8.3 um

um [Kay, 1995] is more a toolkit than a shell system. It provides mechanisms for maintaining a UMKB and offers several tools for accessing and evaluating user model structures. Applications can add further tools that have the same access and evaluation possibilities.

The basic UMKB entity in um is a frame-like structure named "component". Components are organized in a hierarchy of topics and subtopics, which are called "partial models". This structuring is mainly utilized by tools that allow the user to inspect and modify the user model.

um offers three ways for representing components: minimal, basic, and extended. A *minimal* component just consists of a name with perhaps an associated string, thus allowing to represent a simple flag in the UMKB. In a *basic* component there is a name, too. In addition, there are slots for a textual description of the component, for its type, and for its value. Name and description can be used in communication with the user. Using the 'type' slot, components may be labeled as preference, knowledge, belief, or as free attribute. In case of the first three types, a boolean component value is stored in the 'value' slot. This slot is further linked to an evidence list, which contains information about how the component value is justified (by observation, by inference, etc.). Attribute-type components may have non-boolean values, e.g. character strings or numbers, which are stored in the 'extras' slot. This slot is generally provided for information which the application (the "um-consumer" in terms of Kay) wants to lodge in the component. In a minimal component however, it can be used to assign a string to the component name. In an *extended* representation, the 'extras' slot can be employed by an application to hold all additional data that application-specific tools require to be associated with a component.

From a logical point of view, boolean-valued components correspond to positive or negative atomic propositions. Attribute-type components provide attribute-value structures that correspond to ground literals of predicate logic and hence do not exceed propositional logic. Thus, the expressive power of components is quite restricted. However, um concentrates on the use of additional information that is contained in or associated with a component. User modeling in um mainly is about handling evidence information for assumptions about the user, not about handling the assumptions itself. Hence, the central information in a um component is the evidence list that is associated with the 'value' slot.

For basic reasoning capabilities, um provides a set of tools that manipulate the evidence list of a component. That is, reasoning is not used to discover new or implicit assumptions about the user, but to add positive or negative evidence to a representation item. Different (perhaps conflicting) evidences are resolved to a component value when necessary. A single evidence may be due to a statement of the user (e.g., in an interview) or an observation of the application. In addition, there are inference rules that can be labeled as providing rule or stereotype evidence. The antecedent of such a rule is a boolean expression based on component values. If this expression is satisfied, the conclusion will add new pieces of evidence to components. Like in GUMS and UMT (see Section 8.4), rules are meta-level rules of the system. Like in UMT, they are applied in a forward reasoning manner. um rules process component values. They can be compared to implications of propositional calculus even if non-boolean values of attribute-type components are processed. In this case, rule premises will contain relational expressions like 'age > 17', which can be regarded as ground atoms of predicate calculus. There is only one mechanism for both rule inferences and stereotype inferences. Such an approach has also been used in [Beaumont, 1994].

Three assumption types in the sense of AsTRa can be distinguished by um using the type slot of components, namely preferences, knowledge, and beliefs. The fourth possible component type, attribute, is kind of a "non-type", which leaves the interpretation of the component to the application. The difference between knowledge and beliefs is the typical difference between user beliefs that are correct with respect to the system developer's view and those that are not. Hence, the assumption types of um can be compared to the AsTRa types SBUPref, SBMB, and SBUB, respectively. However, assumption types are distinguished component-wise; the UMKB structure of um is kind of orthogonal to that of an AsTRa implementation, which summarizes all UMKB contents of the same type within one partial knowledge base. In particular, the partial models of um do not correspond to assumption type knowledge bases.

Besides reasoning tools, um offers a set of primitives for accessing evidences, components, and partial models. Component values cannot be manipulated directly; an 'add_evidence' function can be used to extend the evidence list of a component. Nevertheless, component values can be retrieved; the application can specify the mechanism that is used to resolve the evidences of the queried component to a value. These mechanisms can be compared with the tell and ask functions of a logic-based user model representation system. In contrast to these, um interface facilities do not deal with UMKB contents, but with evidence information related to UMKB contents.

8.4 UMT

The shell UMT [Brajnik and Tasso, 1992; Brajnik and Tasso, 1994] is similar to um in that it is not the main goal of the system to provide powerful representation and reasoning mechanisms. Its most outstanding feature is its reason maintenance system that allows UMT to revise the user model in a well-defined way. Since the reason maintenance mechanism of UMT is built upon an assumption-based truth maintenance system (ATMS) [de Kleer, 1986], the UMT approach is called "assumption-based user modeling"[1].

Like in GUMS, a user modeling knowledge base consists of the current user model and a set of stereotypes. In contrast to GUMS, which permits tree-like stereotype hierarchies, in UMT stereotypes can be organized in a multiple inheritance network. A user model is a set of attribute-value pairs; the possible attributes and their possible values are predefined by the "generic user stereotype". There are two kinds of values, namely premises and assumptions. Premises are considered to be always true and are usually supplied from sources outside UMT. Assumptions are less reliable values, which may also be reported from outside (e.g., by the user), but mainly stem from stereotypes. Each stereotype defines value assignments for a subset of the possible attributes. A stereotype is activated when a given trigger is satisfied; then, its value assignments are applied to the user model.

Besides stereotypes, the user model developer can define rules and special inconsistency constraints. After stereotype activation, all rules and constraints are applied to the current set of premises and assumptions in order to derive further value assignments and to detect possible inconsistencies. Then, the reason maintenance component determines all possible consistent user models, each of which contains all premises, a set of assumptions and all further assumptions that are derivable from the former. From this collection of user models, the most plausible model is selected, according to a preference relation on the possible sources of information, and then becomes the current user model. The complete process of determining the current user model is inherently complex, which probably is a severe obstacle for the applicability of UMT.

In UMT, there is no distinction of UMKB contents other than the distinction between "premise" and "assumption" attribute values according to their reliability. These categories do not correspond to AsTRa assumption types. In an example application presented in [Brajnik and Tasso, 1994], typical attributes concern user preferences and interests. However, there is no means to explicitly categorize user model contents like, e.g., the type information of um components. Similar to GUMS and um, rules in UMT are not elements of the user model but meta-level rules of the system, which can be compared with type-external expressions of the extended AsTRa framework. Therefore, the expressive power of the

[1] Note that the use of "assumption" as a technical truth maintenance term differs from our use of the word to refer to user model contents.

user model representation formalism of UMT corresponds to ground literals of predicate logic, i.e., it is of propositional complexity.

UMT is designed to become a component of an adaptive application. It features a tell-and-ask interface which allows its application front ends to access the UMKB. Besides entering new premises or assumptions into the user model and retrieving user model contents, UMT applications may give positive or negative feedback to UMT, depending on whether an adaptation step based on a specific assumption was accepted by the user or not. Negative feedback invokes the user model revision process. It must be noted that this feedback feature of UMT is related to the circumstance that domain knowledge bases of UMT applications share attributes and possible attribute values with the user model (as defined in the generic user stereotype). In typical UMT applications, the user reacts to attribute value settings of domain objects that she perceives in her interaction with the system. If these settings depend on the current user model settings, the user reactions can immediately be translated into user model feedback.

8.5 TAGUS

The shell TAGUS [Paiva and Self, 1995] was developed with learner modeling in mind. It maintains a model of the current user (the “User or Learner Model” ULM), which is not explicitly structured, but may contain four kinds of knowledge: characteristics, beliefs, reasoning capabilities, and reasoning strategies of the user. Besides, goals of the user are said to be part of the ULM, but it seems that goals are only used during simulation of the user’s reasoning (see below).

Beliefs and characteristics are represented in a logical language using the special predicates ‘b’ and ‘characteristic’, respectively. A characteristic is denoted by a symbol, while the ‘b’ predicate has first-order logic formulas as arguments. In [Paiva and Self, 1995], the syntax of these formulas is not exactly described, but the presented examples indicate that existential quantifiers are not allowed, and that negation is not possible in the conclusion parts of implications. The latter restriction makes the belief language of TAGUS a bit weaker than the UMKB representation language of GUMS, and significantly weaker than FOPC, which can be used to represent user beliefs in BGP-MS. The belief predicate itself may be negated, which permits the representation of SB$\sim$UB assumptions.

Reasoning capabilities of the user are described as Prolog-like rules, which are called “reasoners”. These rules can be used to reason about beliefs, but also about goals that the user currently pursues. They are employed for simulating problem solving processes of the user, which is particularly useful in tutoring systems.

Reasoning strategies are situated on a meta level. They are represented with so-called “monitors”. Monitors are production rules that describe conditions, on which the user will perform meta-level actions like applying a specific reasoner or giving up the problem solving process. The conditions for monitor application can check current goals as well as (derivable) beliefs of the user.

TAGUS basically provides two kinds of reasoning. First, a *ULM-derivation* is a backward reasoning process that can be employed to check if a user is able to derive something (typically a belief) using her beliefs and reasoners. A *UL-proof* is carried out on the meta-level by processing a given goal using reasoners again, but controlled by the user's reasoning strategies as defined by the available monitors. Monitor conditions may invoke ULM-derivations, so that UL-proofs and ULM-derivations can get nested.

In a TAGUS user model, different kinds of contents can be distinguished. However, these do only partly correspond to assumption types as proposed in the AsTRa framework. User characteristics can be regarded as SB assumptions, while user beliefs and goals in TAGUS clearly are SBUB, SB∼UB, and SBUW assumptions. Syntactically, only the characteristics and belief types are represented explicitly. Reasoners are rules that the user believes to be applicable. On the one hand, they may be regarded as SBUB assumptions. On the other hand, reasoners are not type-internal rules, since they also reason about user goals. So, they could be represented as type-external formulas in the extended AsTRa framework. However, their character as user-held rules would then be lost. It may be considered a weakness of the AsTRa framework that user rules, which reason about assumptions of different types, can only be represented as type-external rules, which cannot be distinguished from system inference rules.

TAGUS provides a rich interface to its UMKB in form of a set of "services", which can be used to access characteristics and beliefs. All services take at least two parameters; the first one specifies whether a characteristic or a belief is dealt with, while the second one is a content expression that specifies a concrete characteristic or belief. TAGUS services very closely resemble the UMKB access functions in the AsTRa framework. The latter take a type-internal expression $AT{:}ac$ as one parameter, which also constitutes a separate specification of assumption type and assumption content information.

In TAGUS, there are two services for querying the user model. One of them, 'ask-ULM', returns a simple answer like 'yes' or 'no', in this respect being similar to the AsTRa **ask** function. The other one, 'inquire-ULM', additionally returns justifications of the answer. Four services are available for updating the user model. 'add-ULM' will just insert a new user model content (like the AsTRa **store** function), while 'tell-ULM' may refuse to insert an expression for reasons of consistency maintenance (similar to AsTRa **tell**), but may also invoke special acquisition rules which in return can add or tell further contents to the ULM. The other two update services can cause user model entries to be revised or even removed. These services require the specification of justifications for their actions. They make use of the reason maintenance system AMMS that is included in TAGUS.

8.6 FITS

FITS [Ikeda and Mizoguchi, 1994] is a framework for intelligent tutoring systems, which has been implemented as an application-independent system. Even more than TAGUS, FITS aims at offering generic services for student modeling. In student modeling, it is even more obvious than in user modeling that the user's (the student's) behavior is typically nonmonotonic. During interaction with an intelligent tutor, the student's knowledge (hopefully) changes, which must be reflected by nonmonotonic processes in the student modeling component. FITS focuses on dealing with nonmonotonicity using quite sophisticated methods.

The language for representing student models in FITS is called SMDL (student model description language). It is similar to the representation formalisms of GUMS and TAGUS in being Prolog-based. Another similarity to GUMS is the use of four different truth values. In contrast to GUMS, however, where truth values become visible as result of default reasoning processes, in SMDL truth values are explicitly represented. A further difference to GUMS is that only three of these values ('true', 'false', and 'unknown') are used to graduate the student's belief in a fact, while the fourth ('fail') is used to express that the modeling system cannot make an assessment of the student concerning a given fact. Thus, 'fail' can be regarded as 'unknown' on the system level. In SMDL, assumption type information is not expressed; it only deals with student beliefs, so that mainly SBUB* assumptions are represented.

SMDL is exclusively used to represent assumptions about the current student, either about her knowledge of facts or about the rules that she is assumed to know and employ in her own reasoning. Then, the student model is regarded as a program written in SMDL, which can be used to derive the truth value that the model currently assigns to a given fact. Thus, the reasoning system of FITS, HSMIS (which is described in detail in [Kono *et al.*, 1994]), supports a backward derivation process, which can be characterized as simulative reasoning and can be compared to the ULM-derivations of TAGUS.

Also meta-level reasoning is done in HSMIS, based not upon explicitly represented knowledge but on an integrated process of diagnosis, induction and truth maintenance. This process builds on so-called oracles, which are pairs of problems (questions asked by the tutoring system) and student answers. The main activity of HSMIS then is to check if a given oracle is consistent with the student model, to identify the cause of that difference (diagnosis) and revise the model by adding or removing rules. Like in UMT, an ATMS is employed to ensure the consistency of the changing student model. The inductive part of this process is that new rules can be generated based on student answers which are uncovered by the current student model. Note that by the way HSMIS works, FITS seems to be well applicable to tutoring systems that use a question/answer approach to tutoring, with a strong guidance of the student, but less suitable to systems with a more explorative learning approach.

HSMIS and the SMDL student model are only parts of the student model module of FITS, and there are further modules in FITS dealing with teaching material (expertise module) and tutoring strategies (tutoring module). Although there must be an interface to both components that other parts of the student module or other modules of FITS use for communication, it is not described in the cited papers.

8.7 Doppelgänger

The system Doppelgänger [Orwant, 1994; Orwant, 1995] was designed as user modeling server. It maintains a number of user models which are constructed from the output of so-called sensors (hardware devices or software modules) that collect information about the user. The user models are made available to arbitrary applications, so that user model contents are no longer private to one application only, as it is the case in most other shells. Besides Doppelgänger, only BGP-MS allows applications to share user models [Pohl and Höhle, 1997] (cf. Section 7.3.3).

Doppelgänger user models are first divided into submodels according to the applicability of user model contents. One example is the 'primary' submodel, which typically contains information about the user that is not likely to change, similar to the "premise" assumptions of UMT. Most submodels focus on a thematical aspect of the user's behavior, like for example a 'news' submodel which contains information about the user's news reading behavior.

Within submodels, there is a further division into assumption types, some of which are obviously similar to AsTRa assumption types, while others are not so closely related to typical AsTRa types. The 'primary' submodel will normally contain 'biographical data', while a 'news' submodel might contain reading 'preferences' and 'times' of user activities. The content formalisms used by Doppelgänger are quite specific to the different assumption types. Attribute-value pairs are used for biographical data, and logical assertions (ground literals) express preferences. Doppelgänger is the only user modeling shell that represents statistical data using assumption types like 'times'; the corresponding assumption contents may be sensored behavior data or estimates about future behavior. In the AsTRa framework, content formalisms can be used for all assumption types. However, it seems plausible that one formalism can be more suitable to a specific assumption type than others.

Doppelgänger does not employ logic-based reasoning mechanisms. Hence, reasoning processes of the user cannot be simulated based on user-held rules, like in TAGUS. Neither can meta-level or acquisition rules be formulated that may be used to infer further assumptions about the user. Doppelgänger uses statistical and machine learning algorithms, which process the observations that are reported by sensors. Similar to um, the main purpose of these mechanisms is to collect evidence information about user behavior and to transform it into strength and confidence values that are associated with individual assumptions. These values represent graduations of Doppelgänger's assumptions about the user. Confidence values represent degrees of system belief, while strength values graduate the user's assumed attitude. In case of preferences, for example, SB*UPref* assumptions are represented. Like in um and in TAGUS, the source of each assumption, i.e., the learning technique that was used to compute the graduation values, is stored. A similar mechanism of storing source information with user model contents has been implemented in BGP-MS (see Section 7.3.2). By using clustering algorithms, so-called "community models" are computed from the available user models. They have the same purpose like stereotypes, namely providing a quick means of acquiring user model contents. However, unlike stereotypes, community models are not predefined.

Since the reasoning mechanisms of Doppelgänger resemble those of um, it is not surprising that also the UMKB interfaces of both shell systems are similar. Like um, Doppelgänger does not provide a real tell interface to applications; its inputs, which are sent by sensors, consist of observations about user behavior that add evidence to user model contents. Moreover, Doppelgänger is similar to um in that it enables users to inspect and modify their models. Applications can retrieve user model contents using a matching mechanism. In queries to Doppelgänger, they specify submodel, assumption type and details of the desired information. Doppelgänger responds with all user model contents that match this specification.

Chapter 9

Summary and Perspectives

9.1 Retrospect and Summary

Initially, there was the task to develop and implement sophisticated representation and reasoning mechanisms into the user modeling shell system BGP-MS. Much thought, conceptual work, and implementation efforts had been investigated into that system before. On the one hand, its design was certainly influenced by the tradition of symbolic knowledge representation, which itself has always (or better, in its best moments) been based upon the solid rocks of logic-based formalisms and methods. On the other hand, it had gone astray from the paths of purely logical approaches by incorporating specialized mechanisms (namely, a partition mechanism and its application to stereotype reasoning), which were of obvious practical use, but were criticized to be ad-hoc and of inferior ability.

In this work, we have taken up early ideas of extending BGP-MS (i.e., its partition-based representation and reasoning facilities) with modal logic. It was our aim to attempt a reconciliation of the partition approach with the modal logic approach to user model representation and reasoning. Our efforts have produced two main results: First, a framework for user model representation and reasoning was introduced that combines the formal semantics and expressive power of modal logic with the flexibility and the potential for integrating specialized mechanisms of a partition-like representation. Second, the representation and reasoning facilities of BGP-MS were significantly extended and improved. As side effects, there are the development of special methods for reasoning with negative assumptions, and, last but not least, the characterization of basic user model contents, which we have called assumptions about the user, as consisting of assumption type and assumption content information. This dichotomy and the corresponding notation for user model contents may become the core of a standard notation for user model contents; we have shown that it can be extended to cover additional aspects of user model contents like uncertainty.

We will now briefly recapitulate the previous chapters. The resulting summary is the starting point for the discussions in the subsequent sections.

In the first part of this thesis, we gave an overview of logic-based user modeling by reviewing both user modeling systems and abstract issues of user model representation and reasoning. In the introduction, we had already discussed the dichotomy of user modeling knowledge base entries into assumption type and assumption content, and that both aspects may be represented on two separate levels. In Chapter 2, we had a closer look at assumption types: The range of possible assumption types was discussed, and several (user modeling) systems with two-level, partition-based representation facilities were reviewed. Based on a comparison of the partition approach and the modal logic approach to user model representation and on a discussion of early integration efforts, we sketched a new proposal for bringing both representation paradigms together, namely the AsTRa framework. Our proposal was particularly concerned with user modeling shell systems and therefore pursued two goals: (1) to provide powerful methods for systems with sophisticated needs, and (2) to offer flexibility, i.e., to be tailorable and hence appropriate for a wide range of applications. Consequently, the subsequent chapters described the logic-based methods that user modeling applications have employed for representation and reasoning purposes. This extensive review showed that the whole range of logic-based methods, from simple propositional atoms to complex modal logic formulas, have been utilized for representation of user modeling knowledge. Moreover, forward- and backward-directed reasoning have played a role in user modeling systems, with stereotype reasoning being a very popular specialization of forward reasoning. The ways of using reasoning techniques could be characterized as primary inferences (based on observations) vs secondary inferences (based on existing user model contents) and as simulative (user) inferences vs meta-level (system) inferences.

The second part dealt with the logic-based representation framework which this thesis proposes particularly for user modeling shells and with its implementation in the shell BGP-MS. In Chapter 5, the AsTRa framework was specified in detail. Its basic mechanisms permit the distinction of an extensible set of assumption types as separate partial knowledge bases. Within an assumption type, different formalisms may be available for specialized representation of assumption contents. Each formalism is required to offer a set of knowledge base access functions, which get used for basic UMKB access including reasoning. Formalisms are further required to be logic-based; if this is the case, then equivalence relationships between basic AsTRa and the simple modal logic AL can be proven. That is, for AsTRa there is a modal logic semantics. On this foundation, the framework was extended by methods for processing the unlimited version of AL, AL^+. Although this full modal logic subsumes the basic mechanisms, it does not replace them. It can be used optionally, and it integrates the UMKB contents that are represented within assumption types. Modal logic can be used to represent negative assumptions; based on the modal logic semantics of AsTRa, however, specialized reasoning procedures for this interesting category of user modeling knowledge were developed. On the whole, AsTRa offers five different reasoning levels, thus offering a wide variety of choices. The whole variety was implemented in BGP-MS, using a partition mechanism for assumption types and also stereotypes, and including SB-ONE and FOPC as content formalisms. Thus, BGP-MS, which was described in Chapter 6, was improved to offer both powerful methods and flexibility of use. Both characteristics were demonstrated in Chapter 7 within an example scenario. In addition, further components of BGP-MS were presented that are related to the representation and reasoning facilities of the shell.

The concluding discussion part has already started with the previous chapter, where we investigated several related systems, mainly other user modeling shells, and compared them with BGP-MS and the AsTRa framework.

We have already hinted at the potential of the AsTRa framework which gets inherited to AsTRa implementations like BGP-MS:

- Powerful, general-purpose mechanisms are integrated with less sophisticated but specialized representation tools. As far as representation is concerned, it is always possible to choose between type-internal representation within (positive or negative) assumption types and type-external representation with modal logic. As to reasoning, a choice of five reasoning levels is provided; in addition to type-internal and full modal logic reasoning, mechanisms for database-like UMKB access and for specialized processing of negative assumptions are available. A crucial point is that, in general, higher levels integrate lower levels.

- The generic approach of AsTRa is most appropriate for user modeling shells. A wide and yet extensible variety of assumption types and stereotypes can be represented. The two-level architecture of AsTRa and the independent treatment of assumption content formalisms allow for the implementation of tailorable tools, or let shell developers constrain the set of implemented mechanisms beforehand.

- Inference mechanisms are offered on four of the five AsTRa reasoning levels (except level 0) by content formalisms, specialized negative type reasoning, and modal logic reasoning. On all levels, forward- and backward-directed reasoning as well as checking UMKB consistency are possible. These options can be utilized for acquisition of (mainly secondary) assumptions about the user in both input- and query-driven ways. Reasoning within assumption types is of a simulative nature since it only concerns beliefs, goals, etc. that are ascribed to the user. Reasoning with modal logic, however, is meta-level reasoning, since type-external modal formulas and axioms constitute meta-level knowledge of the system.

Of course, the AsTRa framework does not solve all user modeling problems. There are several shortcomings, which are mainly due to the focus of this work on logic-based mechanisms and, above all, on the integration of the two main logic-based approaches to user model representation.

- General mechanisms that cope with user model dynamics (i.e., the permanent change that a user model may inherently be subject to) are not available. Techniques that have been applied to this problem in user modeling are nonmonotonic reasoning, reasoning with uncertainty, and reasoning about time. All of these techniques are supported neither by AsTRa nor by BGP-MS. An exception is stereotype reasoning: The retraction of stereotypes in BGP-MS can be regarded as a nonmonotonic reasoning process.
- AsTRa does not explicitly support the acquisition of primary assumptions. BGP-MS offers some useful tools for this purpose. However, it is weak in processing observations about user behavior. This problem is due to the mainly deductive reasoning potential of logic-based formalisms. For acquiring primary assumptions about the user from observations about user behavior, inductive and also abductive reasoning are more promising.
- BGP-MS can be used flexibly, but remains a large-scale system. In principle, smaller, specialized user modeling tools can be useful in various application scenarios. In BGP-MS, modularization of user modeling knowledge was achieved with domain-based user modeling.

These three issues shall be further discussed in the subsequent sections.

9.2 Extensions to Logic-Based User Modeling

This work has dealt with logic-based methods for representation of and for reasoning with user modeling knowledge. There are two issues that are not covered by the developed techniques but which are of central importance to user modeling: the nonmonotonicity of user models and the uncertainty of assumptions about the user. Here, nonmonotonicity means that user models are dynamically changing, and that new observations may lead to assumptions about the user that are in conflict with existing assumptions. So, a user model representation system should be able to behave like nonmonotonic logic formalisms: If a new assumption is added, it is not guaranteed that the conclusions which can be drawn from the user model remain the same. It is even demanded that assumptions can be retracted in favor of other conflicting assumptions which are newer or are otherwise stronger supported. In a representation system that permits inferences, the retraction of an assumption A requires that other assumptions, which were inferred from A, must also be retracted if they are not justified otherwise. In sum, user modeling poses a truth maintenance problem to representation and reasoning systems.

A related problem is the uncertainty of assumptions about the user. [Jameson, 1995] discusses the problem of belief ascription, which is not only an issue in user modeling, but also in other disciplines like psychology and philosophy. Jameson presents several arguments, partly backed by results of experimental psychology, for his case that reasoning about another person's beliefs is reasoning under uncertainty. He discusses the three predominant approaches for ascribing portions of background knowledge to other agents, namely the use of stereotypes, the ascription of one's own knowledge (also known as perturbation, cf. Section 2.3.4), and the application of some kind of default reasoning. Each of these approaches is shown to be inadequate, if a notion of absolute truth is involved, i.e. if only certain ascriptions are made. As psychological studies show, in general there are very few items that can certainly be assumed to be known to a group of people. As to perturbation, people do not ascribe to others those parts of their own knowledge which they think to be difficult or specific. In general, estimates about other person's beliefs seem to depend on the difficulty or specialty of the item under consideration. The notion of difficulty was already employed by Chin in the double-stereotype approach, which he developed for the user modeling component KNOME [Chin, 1986; Chin, 1989]. Jameson extended this approach and developed the theory of intuitive psychometrics [Jameson, 1992] that got employed in the natural language dialog system PRACMA [Jameson *et al.*, 1995]. In this theory, the ascription of a belief to an agent depends on the difficulty of the item under consideration and the knowledgeability of the agent in the area or subdomain the item is related to. The latter magnitude is related to using stereotypes of subdomain experts. In Jameson's work, the three types of belief model contents (i.e., belief, difficulty, and knowledgeability) are all represented as nodes of one coherent probabilistic reasoning network (a so-called Bayesian network); the mentioned dependencies are represented by causal links in the network. In such a representation, the change of a certainty value at one node influences the values of both parent and child nodes. In general, in representations of uncertainty, truth maintenance is achieved by the propagation of values along inferential relationships. The propagation task relates uncertainty representations to truth maintenance approaches of logical representations.

The AsTRa framework for logic-based representation of and reasoning about user modeling knowledge neither includes a truth maintenance method nor does it permit the representation of uncertainty. So, the question may arise, what is it worth then? The answer is that, in principle, uncertainty management and truth maintenance can be added to a system that features logic-based reasoning only. In Section 2.2, a possible approach for combining logical inferences with uncertainty management was sketched. Uncertainty values can be employed to graduate assumptions about the user at any level of nesting of assumption types. If the logical system infers a secondary assumption from other assumptions, an external uncertainty mechanism would compute uncertainty values for the new assumption from the values of the other assumptions. Thus, dependencies between graduation values are created, immediately related to the inferential relations of logical reasoning. These dependencies have to be considered if graduation values are changed from outside. Hence, the logic-based inference process is demanded to trace the inference steps it performs and to finally protocol the relations between concluded knowledge and the premises that were required to make the conclusion. So, a network is established between user model contents.

Such a network can be used for uncertainty propagation, but also for truth maintenance without graduation values. The main aspect of a truth maintenance system are relationships between premises (assumptions) and conclusions, which are managed efficiently. So, if inference traces are made available by a logic-based reasoning system, it can become extended with truth maintenance techniques.

In sum, the AsTRa framework can be regarded as a logic-based core system that is not in conflict with the nonmonotonicity and uncertainty of user models. As a logic-based system, it may be extended with either uncertainty management or truth maintenance. Other extensions are also imaginable, like mechanisms for reasoning about time. [Giangrandi and Tasso, 1997] present such a mechanism which operates in addition to, but independently of, logic-based reasoning methods. As far as reasoning under uncertainty is concerned, also less direct kinds of cooperation with logic-based systems are imaginable. For instance, [Jameson, 1995] employs modal logic knowledge bases and the proof protocols of a modal logic reasoner to dynamically construct parts of the Bayesian networks that are utilized for modeling dialog partners (i.e., users) of the PRACMA system (cf. Section 1.2).

9.3 User Model Acquisition

The AsTRa framework concentrates on representation of user modeling knowledge and reasoning methods that are deductive, get applied within the UMKB, and hence mainly serve to infer secondary assumptions about the user. However, these are not the only tasks of a user modeling system and hence of a user modeling shell system like BGP-MS. There is one user modeling task that may be even more crucial for the success of an adaptive application than representation and deductive reasoning: user model acquisition. The difficulty of acquisition of assumptions about the user increases with the degree of freedom that the user has in her interaction with a computer system or any other device that attempts to be adaptive. The main source of information about the user are the actions that she performs at the user interface. Additional information may be acquired by interviewing the user, but dynamic personalization of system behavior is possible only by taking user actions into account.

BGP-MS offers support for user model acquisition (cf. Section 7.3.1). First, the interview component controls the execution of user interviews and may additionally be employed to display interviews. Second, dialog act type definitions provide a means of describing simple rules for acquiring assumptions from user actions. So, there is a mechanism for action-based acquisition; but dialog act types are only helpful in cases where a one-to-one correspondence between a single action and a set of assumptions exists. In many cases, however, more than one observation is necessary to make an assumption about the user or, what may be equivalent in some systems, to justify a personalization step.

Therefore, many systems that aim at user-adapted interaction make a record of the user-system dialog. Such systems seem to mainly originate from the area of Human-Computer Interaction. An early example is the system X-AiD [Thomas *et al.*, 1987]; it maintained extensive records on several levels of interaction. E.g., execution of system commands was recorded on the command level, while mouse clicks, mouse moves, and key presses were traced on the event level. On all levels, execution time and action parameters were recorded. On the basis of such records, several adaptive features were realized, one example being the detection of macro commands from command level records [Pohl, 1992]. Later systems like Flexcel [Krogsæter *et al.*, 1994] and the plan recognition system HyPlan [Fox *et al.*, 1994] employed similar recording mechanisms. However, instead of voluminous dialog records, much less data can be sufficient for personalized system behavior. A quite recent example is the "adaptive toolbar" of [Debevc *et al.*, 1996]; this system merely counts the number of executions of system commands.

The common characteristic of these systems is that they aggregate observations about user actions into information upon which adaptivity is based. This information is still related to user behavior; it characterizes the user's *usage* of the system. Therefore, this kind of behavior-related information about the user has been termed *usage model* by some researchers [Grunst *et al.*, 1993]. A usage model can be regarded as part of a user model, since "A user model ... contains assumptions on *all* aspects of the user ..." [Wahlster and Kobsa, 1989, p.6; emphasis added by the author]. In contrast to a mere usage model, a full user model additionally contains assumptions that focus on mental attitudes of the user, like beliefs, goals, or preferences. The user modeling system Doppelgänger [Orwant, 1995] (cf. Section 8.7) is an important example of combining usage modeling with other user modeling aspects.

At the beginning of this section, we noted that user actions are the main information source for user model acquisition. Now we have seen that methods for gathering information from user actions were developed in usage modeling systems. Hence, usage modeling systems may seem to provide solutions to the acquisition problem for user modeling in general. However, the difficulty is that these systems completely abstain from "knowledge-based" user modeling. I.e., they normally do not model attitudes of the user like beliefs and goals, which traditionally are represented with logic-based methods and are required by many adaptive systems. So, also in usage modeling, the gap from user actions to such kind of assumptions about the user has not been bridged completely. Nevertheless, a lesson can be learned for knowledge-based user modeling: A dedicated treatment of user actions can lead to relevant information about the user. Such aggregated, action-related information may further lead to knowledge-based, attitude-related assumptions. In the last years, interface agents [Maes, 1994] or personal assistants [Mitchell *et al.*, 1994] were important representatives of usage modeling systems. They use machine learning algorithms to process usage data. Traditionally, there are also machine learning methods that are able to generate symbolic, knowledge-based (domain) models. Hence, they are promising candidates to be employed for the acquisition of knowledge-based user models. They will add *inductive* capabilities to user modeling systems, which, like *abductive* reasoning [Appelt and Pollack, 1992; Stein *et al.*, 1997], will certainly enhance the potential of logic-based, deductive user modeling systems. A first sketch of an architecture for user modeling systems that learn about the user was presented in [Pohl, 1996b].

9.4 User Modeling Modules

The AsTRa framework permits the implementation of application-independent user modeling representation systems that are inherently flexible. Flexibility is mainly due to the two-level representation approach of AsTRa, based on the dichotomy of assumption types and assumption contents. So, in basic AsTRa systems, a choice of content formalisms can be offered. On the assumption type level, specialized inferences can be supported, like inheritance and propagation in BGP-MS. Negative assumption types implement further specialized inference procedures, which can be employed instead of or in combination with powerful but complex modal logic reasoning mechanisms. So, in principle, an AsTRa system is open to modification: the range of content formalisms can be extended, cooperation mechanisms between content formalisms can be added, and on the assumption type level further or different specialized inferences are possible. However, such changes or extensions can be done only by the developers of an AsTRa system like BGP-MS, but not by the developers of the user modeling system who would perhaps like to employ some specific content representation formalism. Particularly on the content level, an interface for adding a content formalism is imaginable. In order to become employed in the system, the language of the formalism (in form of a parsing function) and its access functions must be defined. Furthermore, its mechanisms for maintaining a knowledge base need to be integrated into the assumption type implementation. In BGP-MS, e.g., a general database mechanism is employed for assumption contents of both available formalisms. Therefore, the specification of a new formalism would have to determine how its assumption contents would become integrated into this database. The integration is necessary, since partition inheritance and propagation rely on the database component.

Not only content formalisms, but also other components of an AsTRa system can be regarded as modules, which may be used but may also be omitted. In the case of BGP-MS, two different versions of this shell were put together. The first is a full version, where all representation and reasoning methods are used in an integrated fashion. The second version was tailored to the needs of several user modeling applications; it is a smaller variant with SB-ONE as only content formalism and with negative partitions as only extended mechanism. In principle, the modular design of BGP-MS would permit a fairly free configuration of other variants; a configuration tool was planned for BGP-MS, but could not be realized.

There are two main situations that a system like BGP-MS will be a part of: First, it can be used as user modeling component of a single application. In this case it would be important to tailor the system installation as a whole to the needs of the application. Second, BGP-MS may be employed as a centralized user modeling system that serves several applications or at least several application instances. In this scenario, a full installation of BGP-MS should be available. However, it would be beneficial if each application could specify what components it makes use of. In terms of domain-based user modeling (cf. Section 7.3.3), this specification would ideally be made domain-dependent, since in every user modeling domain there may be different representation and reasoning needs. An even more fine-grained differentiation would be possible, if for every assumption type the use of dedicated mechanisms could be specified. This would be in line with the practice of the user modeling server Doppelgänger [Orwant, 1995] (cf. Section 8.7), where dedicated representation schemes are employed for each type of assumptions.

A modular approach to building user modeling tools was pursued by Kay in her um system [Kay, 1995] (cf. Section 8.3). um is built as a toolkit, i.e. a collection of tools that may be employed by a user modeling application or may not. So, for a single application the necessary user modeling tools can be selected. As far as user model representation is concerned, um has a fixed representation scheme for user model entries, called component. The application may choose to use only minimal parts of a component, although this minimalist representation does not constitute a separate um module. However, the representation tool of um offers interfaces for extensions, e.g. for complex probabilistic methods that may be used to manage the uncertainty values that can be stored in um user models.

9.5 An Outlook

It is difficult to predict the future of user modeling shell or tool systems exactly. On the one hand, there will probably be the need for powerful tools that are employed in centralized settings. Application examples are WWW-based information systems that employ a user modeling system on the server side for adapting to informational needs of site users. Also, like in case of Doppelgänger, a centralized user modeling system is useful if user model contents are to be concerned with the overall working environment of users. On the other hand, there will be a need for small, low-scale user modeling modules with very limited but specialized capabilities. Such modules may be employed not only in computer systems, but may be present in every-day devices like washing machines or television sets. Then, there will be a demand for collections of simple user modeling tools that can be tailored to such a specific, localized application. Of course, a combination of centralized and localized user modeling is also possible. Localized adaptivity modules can contribute their assumptions about the user to the overall user model that a centralized system maintains. However, such a cooperative, distributed user modeling scenario requires standardization with respect to both communication mechanisms and communicated contents. User modeling protocols need to be established in order to make user modeling a ubiquitous process.

Notabene: Such protocols ought to be concerned with the communication between user modeling systems and the user, too. It is a fundamental demand that users must be owners of user models; i.e., at least a reading access to user models must always be possible. So, in addition to user model representation, reasoning, and acquisition, also user model presentation or visualization is an important task of a user modeling system, which should not be the last to be supported by user modeling shells—um [Kay, 1995] and Doppelgänger [Orwant, 1994] did pioneering work in this respect. However, ownership of user models means even more than just user model inspection. Privacy issues play an important role, which are related to problems like security of data transfer or protection against unauthorized access. User modeling tools are required to care about these issues, which are crucial factors for the success of user modeling. After all, in spite of the dangers of collecting perhaps private information about individuals, we will keep on striving for the success of user modeling as a human-centered technology.

Bibliography

[Allen and Miller, 1991] J. F. Allen and B. W. Miller. The RHET System: A Sequence of Self-Guided Tutorials. Technical Report 325, Computer Science Department, The University of Rochester, New York, 1991.

[Allen and Miller, 1993] J. F. Allen and B. W. Miller. The SHOCKER System: A Sequence of Self-Guided Tutorials. Technical Report (Draft), Computer Science Department, The University of Rochester, New York, 1993.

[Allgayer *et al.*, 1992] J. Allgayer, H. J. Ohlbach, and C. Reddig. Modelling Agents with Logic. In *Proc. of the Third International Workshop on User Modeling*, pages 22–34, Dagstuhl, Germany, 1992.

[Appelt and Pollack, 1992] D. E. Appelt and M. E. Pollack. Weighted Abduction for Plan Ascription. *User Modeling and User-Adapted Interaction*, 2(1-2):1–25, 1992.

[Ardissono and Sestero, 1996] L. Ardissono and D. Sestero. Using Dynamic User Models in the Recognition of the Plans of the User. *User Modeling and User-Adapted Interaction*, 5(2):157–190, 1996.

[Baader, 1996] F. Baader. Logik-basierte Wissensrepräsentation. *KI*, (3):8–16, 1996.

[Ballim and Wilks, 1991a] A. Ballim and Y. Wilks. *Artificial Believers.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[Ballim and Wilks, 1991b] A. Ballim and Y. Wilks. Beliefs, Stereotypes and Dynamic Agent Modeling. *User Modeling and User-Adapted Interaction*, 1(1):33–65, 1991.

[Ballim, 1992] A. Ballim. *ViewFinder: A Framework for Representing, Ascribing and Maintaining Nested Beliefs of Interacting Agents.* PhD thesis, Département d'Informatique, Université de Genève, 1992.

[Beaumont, 1994] I. Beaumont. User Modeling in the Interactive Anatomy Tutoring System ANATOM-TUTOR. *User Modeling and User-Adapted Interaction*, 4(1):21–45, 1994.

[Boyle and Encarnacion, 1994] C. Boyle and A. O. Encarnacion. MetaDoc: An Adaptive Hypertext Reading System. *User Modeling and User-Adapted Interaction*, 4(1):1–19, 1994.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman, 1978] R. J. Brachman. A Structural Paradigm for Representing Knowledge. Technical Report 3605, Bolt, Beranek, and Newman Inc., Cambridge, MA, 1978.

[Brajnik and Tasso, 1992] G. Brajnik and C. Tasso. A Flexible Tool for Developing User Modeling Applications with Nonmonotonic Reasoning Capabilities. In *Proc. of the Third International Workshop on User Modeling*, pages 42–66, Dagstuhl, Germany, 1992.

[Brajnik and Tasso, 1994] G. Brajnik and C. Tasso. A Shell for Developing Nonmonotonic User Modeling Systems. *International Journal of Human-Computer Studies*, 40:31–62, 1994.

[Brownston *et al.*, 1985] L. Brownston et al. *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming.* Addison-Wesley, 1985.

[Carberry, 1989] S. Carberry. Plan Recognition and Its Use in Understanding Dialog. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 133–162. Springer, Berlin, Heidelberg, 1989.

[Chin, 1986] D. N. Chin. User Modelling in UC, the UNIX Consultant. In *Proc. of CHI'86*, pages 24–28, 1986.

[Chin, 1989] D. N. Chin. KNOME: Modeling What the User Knows in UC. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 74–107. Springer, Berlin, Heidelberg, 1989.

[Chin, 1993] D. N. Chin. Acquiring User Models. *Artificial Intelligence Review*, 7:185–197, 1993.

[Christaller *et al.*, 1992] Th. Christaller, F. di Primio, U. Schnepf, and A. Voss, editors. *The AI Workbench Babylon: An Open and Portable Development Environment for Expert Systems.* Academic Press, London, 1992.

[Clark and Marshall, 1981] H. Clark and C. R. Marshall. Definite Reference and Mutual Knowledge. In A. K. Joshi, I. A. Sag, and B. L. Webber, editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, Cambridge, 1981.

[Cohen *et al.*, 1991] R. Cohen, F. Song, B. Spencer, and P. van Beek. Exploiting Temporal and Novel Information from the User in Plan Recognition. *User Modeling and User-Adapted Interaction*, 1(2):125–148, 1991.

[Cohen, 1978] P. R. Cohen. On Knowing What to Say: Planning Speech Acts. Technical Report 118, Department of Computer Science, University of Toronto, Canada, 1978.

[de Kleer, 1986] J. de Kleer. An Assumption-Based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[De Rosis *et al.*, 1992] F. De Rosis, S. Pizzutilo, A. Russo, D. C. Berry, and F. J. Nicolau Molina. Modeling the User Knowledge by Belief Networks. *User Modeling and User-Adapted Interaction*, 2(4):367–388, 1992.

[Debevc *et al.*, 1996] M. Debevc, B. Meyer, D. Donlagic, and R. Svecko. Design and Evaluation of an Adaptive Icon Toolbar. *User Modeling and User-Adapted Interaction*, 6(1):1–21, 1996.

[Fagin *et al.*, 1995] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, 1995.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 5(2):189–208, 1971.

[Finin *et al.*, 1993] T. Finin et al. Specification of the KQML Agent-Communication Language (Draft). `http://www.cs.umbc.edu/kqml/kqmlspec.ps`, 1993.

[Finin *et al.*, 1994] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Third International Conference on Knowledge and Information Management*, pages 456–463, New York, NY, November 1994. ACM Press.

[Finin, 1989] T. W. Finin. GUMS: A General User Modeling Shell. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 411–430. Springer, Berlin, Heidelberg, 1989.

[Fink and Herrmann, 1993] J. Fink and M. Herrmann. KN-PART: Ein Verwaltungssystem zur Benutzermodellierung mit prädikatenlogischer Wissensrepräsentation. WIS-Memo 5, AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1993.

[Fox *et al.*, 1994] Th. Fox, G. Grunst, and K.-J. Quast. HyPLAN: A Context-Sensitive Hypermedia Help System. In R. Oppermann, editor, *Adaptive User Support*, pages 126–193. Lawrence Erlbaum Associates, 1994.

[Gabbay and Ohlbach, 1992] D. Gabbay and H. J. Ohlbach. Quantifier Elimination in Second-Order Predicate Logic. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference (KR'92)*, pages 425–435. Morgan Kaufmann, San Mateo, CA, 1992.

[Giangrandi and Tasso, 1997] P. Giangrandi and C. Tasso. Managing Temporal Knowledge in Student Modeling. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 415–426, Wien, New York, 1997. Springer-Verlag.

[Goodman and Litman, 1992] B. A. Goodman and D. J. Litman. On the Interaction between Plan Recognition and Intelligent Interfaces. *User Modeling and User-Adapted Interaction*, 2(1-2):55–82, 1992.

[Gori *et al.*, 1997] M. Gori, M. Maggini, and E. Martinelli. Web-Browser Access Through Voice Input and Page Interest Prediction. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 17–19, Wien, New York, 1997. Springer-Verlag.

[Grunst *et al.*, 1993] G. Grunst, R. Oppermann, and C. G. Thomas. Benutzungsmodellierung bei kontext-sensitiver Hilfe und adaptiver Systemgestaltung. In A. Kobsa and W. Pohl, editors, *Arbeitspapiere des Workshops "Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen"*. WIS-Memo 7, pages 69–77. AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1993.

[Haas, 1986] A. Haas. A Syntactic Theory of Belief and Action. *Artificial Intelligence*, 28:245–292, 1986.

[Han, 1995] H. Han. Untersuchung von Alternativen zur Realisierung der logischen Inferenzen in BGP-MS. Diplomarbeit, AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1995.

[Hendrix, 1975] G. G. Hendrix. Expanding the Utility of Semantic Networks through Partitioning. In *Proc. of the 4th International Joint Conference on Artificial Intelligence*, 1975.

[Hendrix, 1979] G. Hendrix. Encoding Knowledge in Partitioned Networks. In N. V. Findler, editor, *Associative Networks, Representation and Use Of Knowledge by Computers*. Academic Press, New York, 1979.

[Hilbert and Ackermann, 1972] D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik*. Springer-Verlag, Berlin, Heidelberg, New York, 6th edition, 1972.

[Hintikka, 1962] J. Hintikka. *Knowledge and Belief.* Cornell University Press, New York, 1962.

[Hintikka, 1969] J. Hintikka. Semantics for Propositional Attitudes. In J. Davis et al., editors, *Philosophical Logic*. Reidel, Dordrecht, 1969.

[Huang *et al.*, 1991] X. Huang, G. I. McCalla, J. E. Greer, and E. Neufeld. Revising Deductive Knowledge and Stereotypical Knowledge in a Student Model. *User Modeling and User-Adapted Interaction*, 1(1):87–115, 1991.

[Hustadt, 1994] U. Hustadt. A Multi-Modal Logic for Stereotyping. In *Proceedings of the Fourth International Conference on User Modeling*, pages 87–92, 1994.

[Hustadt, 1995] U. Hustadt. Introducing Epistemic Operators into a Description Logic. In A. Laux and H. Wansing, editors, *Knowledge and Belief in Philosophy and Artificial Intelligence*, Logica Nova, pages 65–86. Akademie Verlag, Berlin, 1995.

[Ikeda and Mizoguchi, 1994] M. Ikeda and R. Mizoguchi. FITS: A Framework for ITS—A Computational Model of Tutoring. *Journal of Artificial Intelligence in Education*, 5(3):319–348, 1994.

[Jackson and Reichgelt, 1989] P. Jackson and H. Reichgelt. A General Proof Method for Arbitrary Modal Predicate Logic. In P. Jackson, H. Reichgelt, and F. van Harmelen, editors, *Logic-Based Knowledge Representation.* MIT Press, Cambridge, MA, 1989.

[Jameson *et al.*, 1995] A. Jameson, R. Schäfer, J. Simons, and Th. Weis. Adaptive Provision of Evaluation-Oriented Information: Tasks and Techniques. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montréal*, San Mateo, CA, 1995. Morgan Kaufmann.

[Jameson, 1992] A. Jameson. Generalizing the Double-Stereotype Approach: A Psychological Perspective. In *UM92 – Third International Workshop on User Modeling*, pages 69–83, 1992.

[Jameson, 1995] A. Jameson. Logic Is Not Enough: Why Reasoning About Another Person's Beliefs Is Reasoning Under Uncertainty. In A. Laux and H. Wansing, editors, *Knowledge and Belief in Philosophy and Artificial Intelligence.* Akademie Verlag, Berlin, 1995.

[Jameson, 1996] A. Jameson. Numerical Uncertainty Management in User and Student Modeling: An Overview of Systems and Issues. *User Modeling and User-Adapted Interaction*, 5(3-4):193–251, 1996.

[Jennings and Higuchi, 1993] A. Jennings and H. Higuchi. A User Model Neural Network for a Personal News Service. *User Modeling and User-Adapted Interaction*, 3(1):1–25, 1993.

[Kass and Finin, 1988] R. Kass and T. Finin. Modeling the User in Natural Language Systems. *Computational Linguistics*, 14(3):5–22, 1988.

[Kass, 1991] R. Kass. Building a User Model Implicitly from a Cooperative Advisory Dialog. *User Modeling and User-Adapted Interaction*, 1(3):203–258, 1991.

[Kautz, 1991] H. A. Kautz. A Formal Theory of Plan Recognition and its Implementation. In *Reasoning about Plans*, pages 69–125. Morgan Kaufmann, San Mateo, CA, 1991.

[Kay, 1993] J. Kay. Reusable Tools for User Modeling. *Artificial Intelligence Review*, 7(3-4):241–251, 1993.

[Kay, 1995] J. Kay. The um Toolkit for Reusable, Long Term User Models. *User Modeling and User-Adapted Interaction*, 4(3):149–196, 1995.

[Kobsa and Pohl, 1995] A. Kobsa and W. Pohl. The User Modeling Shell System BGP-MS. *User Modeling and User-Adapted Interaction*, 4(2):59–106, 1995.

[Kobsa and Wahlster, 1989] A. Kobsa and W. Wahlster, editors. *User Models in Dialog Systems*. Springer, Berlin, Heidelberg, 1989.

[Kobsa *et al.*, 1994] A. Kobsa, D. Müller, and A. Nill. KN-AHS: An Adaptive Hypertext Client of the User Modeling System BGP-MS. In *Proc. of the Fourth International Conference on User Modeling*, pages 99–105, Hyannis, MA, 1994.

[Kobsa, 1984] A. Kobsa. Three Steps in Constructing Mutual Belief Models from User Assertions. In *Proc. of the 6 th ECAI*, pages 423–426, Pisa, Italy, 1984.

[Kobsa, 1985] A. Kobsa. *Benutzermodellierung in Dialogsystemen*. Springer-Verlag, Berlin, Heidelberg, 1985.

[Kobsa, 1989] A. Kobsa. A Taxonomy of Beliefs and Goals for User Models in Dialog Systems. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 52–68. Springer, Berlin, Heidelberg, 1989.

[Kobsa, 1990] A. Kobsa. Modeling The User's Conceptual Knowledge in BGP-MS, a User Modeling Shell System. *Computational Intelligence*, 6:193–208, 1990.

[Kobsa, 1991] A. Kobsa. Utilizing Knowledge: The Components of the SB-ONE Knowledge Representation Workbench. In J. Sowa, editor, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*, pages 457–486. Morgan Kaufmann, San Mateo, CA, 1991.

[Kobsa, 1992] A. Kobsa. Towards Inferences in BGP-MS: Combining Modal Logic and Partition Hierarchies for User Modeling. In *Proceedings of the Third International Workshop on User Modeling*, pages 35–41, Dagstuhl, Germany, 1992.

[Kono *et al.*, 1994] Y. Kono, M. Ikeda, and R. Mizoguchi. THEMIS: A Nonmonotonic Inductive Student Modeling System. *Journal of Artificial Intelligence in Education*, 5(3), 1994.

[Konolige and Pollack, 1989] K. Konolige and M. Pollack. Ascribing Plans To Agents: Preliminary Report. In *11th International Joint Conference on Artificial Intelligence*, pages 924–930, Detroit, MI, 1989.

[Konolige, 1982] K. Konolige. A First-Order Formalization of Knowledge and Action For a Multi-Agent Planning System. In J. E. Hayes, D. Michie, and Y.-H. Pao, editors, *Machine Intelligence*, volume 10, pages 41–72. Halsted Press, New York, 1982.

[Kripke, 1963] S. Kripke. Semantic Considerations on Modal Logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[Krogsæter *et al.*, 1994] M. Krogsæter, R. Oppermann, and C. G. Thomas. A User Interface Integrating Adaptability and Adaptivity. In R. Oppermann, editor, *Adaptive User Support*. Lawrence Erlbaum Associates, 1994.

[Lehman and Carbonell, 1989] J. F. Lehman and J. G. Carbonell. Learning the User's Language: A Step Towards Automated Creation of User Models. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 161–194. Springer-Verlag, Berlin, Heidelberg, 1989.

[Lewis and Langford, 1932] C. I. Lewis and C. H. Langford. *Symbolic Logic*. Dover Publications, New York, 1932. 2nd edition 1959.

[Lindner and Bodendorf, 1993] H.-G. Lindner and F. Bodendorf. Ein neuronales Konzept für adaptive Anwendungen. In A. Kobsa and W. Pohl, editors, *Arbeitspapiere des Workshops "Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen"*. WIS-Memo 7. AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1993.

[Maes, 1994] P. Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7):31–40, July 1994.

[McArthur, 1988] G. L. McArthur. Reasoning About Knowledge and Belief: A Survey. *Computational Intelligence*, 4:223–243, 1988.

[McCauley *et al.*, 1980] C. McCauley, C. L. Stitt, and M. Segal. Stereotyping: From Prejudice to Prediction. *Psychological Bulletin*, 87:195–208, 1980.

[McCoy, 1989] K. F. McCoy. Highlighting a User Model to Respond to Misconceptions. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 231–254. Springer-Verlag, Berlin, Heidelberg, 1989.

[McCune, 1994] W. W. McCune. OTTER 3.0 Reference Manual and Guide. Technical Report ANL-94/6, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, 1994.

[McTear, 1993] M. F. McTear. User Modelling for Adaptive Computer Systems: a Survey. *Artificial Intelligence Review*, 7(3-4):157–184, August 1993.

[Miller, 1991] B. W. Miller. The Rhetorical Knowledge Representation System Reference Manual. Technical Report 326, Computer Science Department, The University of Rochester, New York, 1991.

[Minker, 1988] J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 1988.

[Minsky, 1975] M. Minsky. A Framework for Representing Knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[Mitchell *et al.*, 1994] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a Learning Personal Assistant. *Communications of the ACM*, 37(7):81–91, July 1994.

[Moore and Paris, 1992] J. D. Moore and C. L. Paris. Exploiting User Feedback to Compensate for the Unreliability of User Models. *User Modeling and User-Adapted Interaction*, 2(4):331–365, 1992.

[Moore, 1980] R. Moore. *Reasoning About Knowledge and Action*. PhD thesis, MIT, Cambridge MA, 1980.

[Moore, 1985] R. Moore. A Formal Theory of Knowledge and Action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Common Sense World*. Ablex, Norwood, NJ, 1985.

[Morik, 1989] K. Morik. User Models and Conversational Settings: Modeling the User's Wants. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 364–385. Springer-Verlag, Berlin, Heidelberg, 1989.

[Niem *et al.*, 1993] L. Niem, B. J. Fugère, P. Rondeau, and R. Tremblay. Defining the Semantics of Extended Genetic Graphs. *User Modeling and User-Adapted Interaction*, 3(2):107–153, 1993.

[Nonnengart, 1992] A. Nonnengart. First-Order Modal Logic Theorem Proving and Standard PROLOG. Arbeitsbericht MPI-I-92-228, Max-Planck-Institut für Informatik, Saarbrücken, 1992.

[Norman and Draper, 1986] D. A. Norman and S. W. Draper, editors. *User Centered System Design: New Perspectives on Human-Computer Interaction.* Erlbaum, Hillsdale, NJ, 1986.

[Nwana, 1991] H. S. Nwana. User Modelling and User Adapted Interaction in an Intelligent Tutoring System. *User Modeling and User-Adapted Interaction*, 1(1):1–32, 1991.

[Ohlbach, 1991] H. J. Ohlbach. Semantics-Based Translation Methods for Modal Logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[Oppermann, 1994] R. Oppermann, editor. *Adaptive User Support.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[Orwant, 1994] J. Orwant. Apprising the User of User Models: Doppelgänger's Interface. In *Proc. of the Fourth International Conference on User Modeling*, pages 151–156, Hyannis, MA, 1994.

[Orwant, 1995] J. Orwant. Heterogeneous Learning in the Doppelgänger User Modeling System. *User Modeling and User-Adapted Interaction*, 4(2):107–130, 1995.

[Owsnicki-Klewe *et al.*, 1993] B. Owsnicki-Klewe, K. v. Luck, and B. Nebel. Wissensrepräsentation und Logik – Eine Einführung. In G. Görz, editor, *Einführung in die künstliche Intelligenz*, chapter 1.1, pages 3–54. Addison-Wesley, Bonn, 1993.

[Paiva and Self, 1995] A. Paiva and J. Self. TAGUS – A User and Learner Modeling Workbench. *User Modeling and User-Adapted Interaction*, 4(3):197–226, 1995.

[Paris, 1989] C. Paris. The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 133–162. Springer, Berlin, Heidelberg, 1989.

[Patil *et al.*, 1992] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, Th. Gruber, and R. Neches. The DARPA Knowledge Sharing Effort: Progress Report. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference (KR'92)*, San Mateo, CA, 1992. Morgan Kaufmann.

[Peter and Rösner, 1994] G. Peter and D. Rösner. User-Model-Driven Generation of Instructions. *User Modeling and User-Adapted Interaction*, 3(4):289–319, 1994.

[Pohl and Höhle, 1997] W. Pohl and J. Höhle. Mechanisms for Flexible Representation and Use of Knowledge in User Modeling Shell Systems. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 403–414, Wien, New York, 1997. Springer-Verlag.

[Pohl *et al.*, 1995] W. Pohl, A. Kobsa, and O. Kutter. User Model Acquisition Heuristics Based on Dialogue Acts. In *International Workshop on the Design of Cooperative Systems*, pages 471–486, Antibes-Juan-les-Pins, France, 1995. INRIA.

[Pohl, 1992] W. Pohl. Beispielerkennung für die induktive Generierung von Benutzermakros. GMD-Studien 205, GMD, St. Augustin, 1992.

[Pohl, 1996a] W. Pohl. Combining Partitions and Modal Logic for User Modeling. In D. M. Gabbay and H. J. Ohlbach, editors, *Practical Reasoning: Proceedings of the International Conference on Formal and Applied Practical Reasoning*, pages 480–494, Berlin, Heidelberg, 1996. Springer.

[Pohl, 1996b] W. Pohl. Learning about the User – User Modeling and Machine Learning. In J. Herrmann and V. Moustakis, editors, *Proc. ICML'96 Workshop "Machine Learning meets Human-Computer Interaction"*, pages 29–40, 1996.

[Quilici, 1989] A. Quilici. AQUA: A System that Detects and Responds to User Misconceptions. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*. Springer, Berlin, Heidelberg, 1989.

[Quilici, 1994] A. Quilici. Forming User Models by Understanding User Feedback. *User Modeling and User-Adapted Interaction*, 3(4):321–358, 1994.

[Quillian, 1968] M. Quillian. Semantic Memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. MIT Press, Cambridge, MA, 1968.

[Reichgelt, 1989] H. Reichgelt. Logics for Reasoning About Knowledge and Belief. *The Knowledge Engineering Review*, 4(2):119–139, 1989.

[Retz-Schmidt, 1991] G. Retz-Schmidt. Recognizing Intentions, Interactions, and Causes of Plan Failures. *User Modeling and User-Adapted Interaction*, 1(2):173–202, 1991.

[Rich, 1979] E. Rich. User Modeling via Stereotypes. *Cognitive Science*, 3:329–354, 1979.

[Rich, 1983] E. Rich. Users are Individuals: Individualizing User Models. *Journal of Man-Machine Studies*, 18:199–214, 1983.

[Rich, 1989] E. Rich. Stereotypes and User Modeling. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 35–51. Springer, Berlin, Heidelberg, 1989.

[Robinson, 1965] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

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

[Sarner and Carberry, 1992] M. Sarner and S. Carberry. Generating Tailored Definitions Using a Multifaceted User Model. *User Modeling and User-Adapted Interaction*, 2(3):181–210, 1992.

[Schäfer and Weyrath, 1997] R. Schäfer and Th. Weyrath. Assessing Temporally Variable User Properties With Dynamic Bayesian Networks. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 377–388, Wien, New York, 1997. Springer-Verlag.

[Schauer and Pohl, 1997] H. Schauer and W. Pohl. Generating Secondary Assumptions in BGP-MS. In R. Schäfer and M. Bauer, editors, *ABIS-97: 5. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen.* SFB 378, Memo 65, Saarbrücken, September 1997. Universität des Saarlandes.

[Schauer, 1998] H. Schauer. First-Order Logic and Secondary Assumptions in the User Modeling Shell System BGP-MS. Master's thesis, Department of Computer Science, University of Koblenz, 1998.

[Scherer, 1990] J. Scherer. SB-PART: Ein Partitionsverwaltungssystem für die Wissensrepräsentationssprache SB-ONE. Memo 48, Projekt XTRA, Fachbereich Informatik, Universität Saarbrücken, 1990.

[Schreck, 1995] J. Schreck. Konzeption der Erweiterung von partitionenorientierter Wissensrepräsentation um modallogische Inferenzen. Diplomarbeit, AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1995.

[Schwab, 1989] Th. Schwab. *Methoden zur Dialog- und Benutzermodellierung in adaptiven Computersystemen.* PhD thesis, Institut für Informatik der Universität Stuttgart, 1989.

[Searle, 1969] J. R. Searle. *Speech Acts.* Cambridge University Press, 1969.

[Shifroni and Shanon, 1992] E. Shifroni and B. Shanon. Interactive User Modeling: An Integrative Explicit-Implicit Approach. *User Modeling and User-Adapted Interaction*, 2(4):287–330, 1992.

[Shortliffe and Buchanan, 1975] E. Shortliffe and B. Buchanan. A Model of Inexact Reasoning in Medicine. *Math. Bioscience*, 23:351–379, 1975.

[Simon, 1995] R. Simon. Realisierung der Erweiterung von partitionenorientierter Wissensrepräsentation um modallogische Inferenzen. Diplomarbeit, AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1995.

[Sleeman, 1985] D. Sleeman. UMFE: A User Modelling Front-End Subsystem. *International Journal of Man-Machine Studies*, 23:71–88, 1985.

[Stein *et al.*, 1997] A. Stein, J. A. Gulla, and U. Thiel. Making Sense of Users' Mouse Clicks: Abductive Reasoning and Conversational Dialogue Modeling. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, Wien, New York, 1997. Springer-Verlag.

[Sukaviriya and Foley, 1993] P. Sukaviriya and D. Foley. A Built-in Provision for Collecting Individual Task Usage Information in UIDE: the User Interface Design Environment. In M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, editors, *Adaptive User Interfaces: Principles and Practise*, pages 197–221. North Holland Elsevier, Amsterdam, 1993.

[Tattersall, 1992] C. Tattersall. Generating Help for Users of Application Software. *User Modeling and User-Adapted Interaction*, 2(3):211–248, 1992.

[Taylor *et al.*, 1996] J. A. Taylor, J. Carletta, and C. Mellish. Requirements for Belief Models in Cooperative Dialogue. *User Modeling and User-Adapted Interaction*, 6(1):23–68, 1996.

[Thomas *et al.*, 1987] C. G. Thomas, G. M. Kellermann, and H.-W. Hein. X-AiD: An adaptive and knowledge-based human-computer interface. In H.-J. Bullinger and B. Shackel, editors, *Proc. of Human-Computer Interaction INTERACT'87*, pages 1075–1080, Amsterdam, 1987. Elsevier Science Publishers.

[van Arragon, 1991] P. van Arragon. Modeling Default Reasoning Using Defaults. *User Modeling and User-Adapted Interaction*, 1(3):259–288, 1991.

[van Benthem, 1984] J. van Benthem. Correspondence Theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 167–247. D. Reidel Publishing Company, Dordrecht, 1984.

[Wahlster and Kobsa, 1989] W. Wahlster and A. Kobsa. User Models in Dialog Systems. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 4–34. Springer, Berlin, Heidelberg, 1989.

[Wallen, 1987] L. Wallen. Matrix Proof Methods for Modal Logic. In *10th International Joint Conference on Artificial Intelligence*, pages 917–923, 1987.

[Webb and Kuzmycz, 1996] G. I. Webb and M. Kuzmycz. Feature Based Modelling: A Methodology for Producing Coherent, Consistent, Dynamically Changing Models of Agent's Competencies. *User Modeling and User-Adapted Interaction*, 5(2):117–150, 1996.

[Weida and Litman, 1992] R. Weida and D. Litman. Terminological Reasoning with Constraint Networks and an Application to Plan Recognition. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference (KR'92)*, pages 282–293. Kaufmann, San Mateo, CA, 1992.

[Wilensky *et al.*, 1984] R. Wilensky, Y. Arens, and D. N. Chin. Talking to UNIX in English. *Communications of the ACM*, 27:574–593, 1984.

[Wilensky *et al.*, 1986] R. Wilensky et al. UC: A Progress Report. Technical Report UCB/CSD 87/303, Division of Computer Science, University of California, Berkeley, CA, 1986.

[Wos *et al.*, 1965] L. Wos, D. Carson, and G. Robinson. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *Journal of the ACM*, 12:536–541, 1965.

[Wu, 1991] D. Wu. Active Acquisition of User Models: Implications for Decision-Theoretic Dialog Planning and Plan Recognition. *User Modeling and User-Adapted Interaction*, 1(2):149–172, 1991.

[Ye, 1997] N. Ye. Neural Networks Approach to User Modeling and Intelligent Interface: A Review and Reappraisal. *International Journal of Human-Computer Interaction*, 9(1):3–23, 1997.

[Yoshida and Motoda, 1996] K. Yoshida and H. Motoda. Automated User Modeling for Intelligent Interface. *International Journal of Human-Computer Interaction*, 8(3):237–258, 1996.

[Zimmermann, 1994] J. Zimmermann. Hybride Wissensrepräsentation in BGP-MS: Integration der Wissensverarbeitung von SB-ONE und OTTER. WIS-Memo 12, AG Wissensbasierte Informationssysteme, Informationswissenschaft, Universität Konstanz, 1994.

[Zukerman and McConachy, 1993] I. Zukerman and R. McConachy. Consulting a User Model to Address a User's Inferences during Content Planning. *User Modeling and User-Adapted Interaction*, 3(2):155–185, 1993.

Index