User Modeling and User-Adapted Interaction 4(2), 59-106, 1995.

The User Modeling Shell System BGP-MS

Alfred Kobsa and Wolfgang Pohl
WG Knowledge-Based Information Systems
Department of Information Science
University of Konstanz
D-78434 Konstanz, GERMANY
{kobsa,pohl}@inf-wiss.uni-konstanz.de

Abstract

BGP-MS is a user modeling shell system that can assist interactive software systems in adapting to their current users by taking the users' presumed knowledge, beliefs, and goals into account. It offers applications several methods for communicating observations concerning the user to BGP-MS, and for obtaining information on currently held assumptions about the user from BGP-MS. It provides a choice of two integrated formalisms for representing beliefs and goals, and includes several types of inferences for drawing additional assumptions based on an initial interview, observed user actions, and stereotypical knowledge about pre-defined user subgroups.

BGP-MS is a customizable software system that is independent from applications, operates concurrently with them, and interacts with them through inter-process communication. For tailoring BGP-MS to a specific application domain, the developer must select those components of BGP-MS that are needed in this domain and fill them with relevant domain-dependent user modeling knowledge.

This paper first summarizes the user modeling services that BGP-MS provides to application programs at runtime. It discusses the representational and inferential foundations that determine the scope and the limits of these services, and also gives a detailed example illustrating the interaction between the various system components. It describes interfaces that are available to application developers for tailoring BGP-MS to the specific user modeling needs of their application domains. Finally, it compares the system with all other major user modeling shell systems, and describes a first application that employs BGP-MS for adapting hypertext to users' terminological knowledge.

Keywords: user modeling shell system, belief and goal modeling, belief and goal representation, hybrid representation, stereotypes, adaptive hypertext, adaptive information presentation

Contents

1 Introduction

Adding user modeling abilities to application systems for catering to the individual needs of users is often quite expeditous. Application developers with this goal in mind need appropriate mechanisms that

- draw assumptions about the user based on his interaction with the application system,
- represent and store these assumptions,
- draw additional assumptions based on initial assumptions,
- take appropriate actions when inconsistencies betweeen assumptions are detected, and
- supply the application with current assumptions about the user.

Until recently, hardly any software tools have been available that facilitate the development of user modeling systems. In most cases, application developers had to program the required user modeling functionality from scratch. During the last few years, however, considerable experience has been gained on what kind of abilities are frequently demanded from user modeling systems. It therefore makes sense to condense basic abilities which most user modeling components must have into "empty", application-independent user modeling *shell systems*. At the time when an application system is developed, a shell system must be selected and filled with user modeling knowledge pertaining to the particular application domain. At runtime, the "filled" shell system will then perform the role of the user modeling system in this application (or at least a large part of it). A similar line of research can be found in the field of expert systems, in which experience gained from individual expert systems lead to the development of expert system shells.

BGP-MS[1] is a user modeling shell that can assist interactive software systems in adapting to users based on assumptions about their knowledge, beliefs[2] and goals. At runtime when the user interacts with the application, BGP-MS accepts information from the application concerning observed beliefs, goals and actions of the user. It can also directly obtain information from the user via a questionnaire. Information about the beliefs and goals of the current user that are communicated to BGP-MS become represented in the user's individual user model. Additional assumptions are acquired by various kinds of inferences that are based on the current entries in the user model, the user's answers in the questionnaire, observed user actions as communicated by the application system, and stereotypical knowledge about user subgroups.

At the time when the application system is being developed, those components of BGP-MS that are needed in the particular application domain must be selected and filled with domain-dependent user modeling knowledge. For instance, the application developer can tell BGP-MS the concepts that are

[1] BGP-MS stands for Belief, Goal and Plan Maintenance System.

[2] The terms 'knowledge' and 'belief' are used here from the perspective of the system. That is, knowledge of the user is regarded as beliefs of the user that are shared by the system. We will therefore only use the more general notion of 'belief' from now on, unless a distinction is relevant.

used in the application domain, so that it can later make assumptions as to which of them are known by the current user. The developer can inform BGP-MS about subgroups among the prospective user population and what their distinguishing characteristics as well as their presumptive beliefs and goals are, so that BGP-MS can later classify the current user as belonging to one or more of these subgroups. The developer can also tell BGP-MS which beliefs and goals the user should be assumed to hold when specific answers are given in a questionnaire, when specific user actions are observed by the application system, or when specific assumptions about his beliefs and goals are already being maintained. Important inferred assumptions that should be immediately signaled to the application can be specified. Software tools and graphical interfaces that facilitate these selection and knowledge specification tasks for the application developer are available as well.

Although it is not yet possible to assess the usefulness of these services on empirical grounds, care was taken that the needs of applications published in the user modeling literature were taken into account (see [Kobsa, 1990]). Moreover, several systems (namely UMFE [Sleeman, 1985], TAILOR [Paris, 1989], ASSIST [Uehara, 1989], EUROHELP [Winkels, 1990] and GENIE [Wolz, 1992]) were analyzed in detail to compare their representational and inferential needs with the respective services provided by BGP-MS [Kleiber, 1994].

The following section describes the kinds of information about the user that BGP-MS can accept from the application system, and discusses how the application can ask BGP-MS for the current assumptions about the user or become automatically notified when certain important assumptions have been made. Section 3 describes the representation systems that BGP-MS employs for storing its models of individual users, its assumptions about characteristics of user subgroups, and its inference rules. Section 4 describes the kinds of inferences that BGP-MS draws when specific answers are given in the user questionnaire, when specific user actions are observed by the application system and communicated to BGP-MS, or when specific assumptions about his beliefs and goals are already being held. Section 5 illustrates the operation of BGP-MS in a comprehensive example and Section 6 describes the interfaces that facilitate the usage of BGP-MS for the user model developer. Section 7 compares BGP-MS with other user modeling shell systems, and Section 8 describes a first application which employs BGP-MS for adapting hypertext to the user's presumed domain knowledge.

2 Information flow between the application and BGP-MS

2.1 Overview

BGP-MS is an independent software system that can communicate with an application system via several types of messages, including the following (see Fig. 1 for examples):

1. The application can inform BGP-MS about observed user beliefs and goals.
2. BGP-MS can send the application interview questions that should be posed to the user.
3. The application can send BGP-MS the user's answers to these questions.
4. The application can inform BGP-MS about observed user actions.
5. The application can ask BGP-MS for its current assumptions about the user.
6. BGP-MS can answer such questions of the application.
7. BGP-MS can unsolicitedly signal important events in the user model to the application.

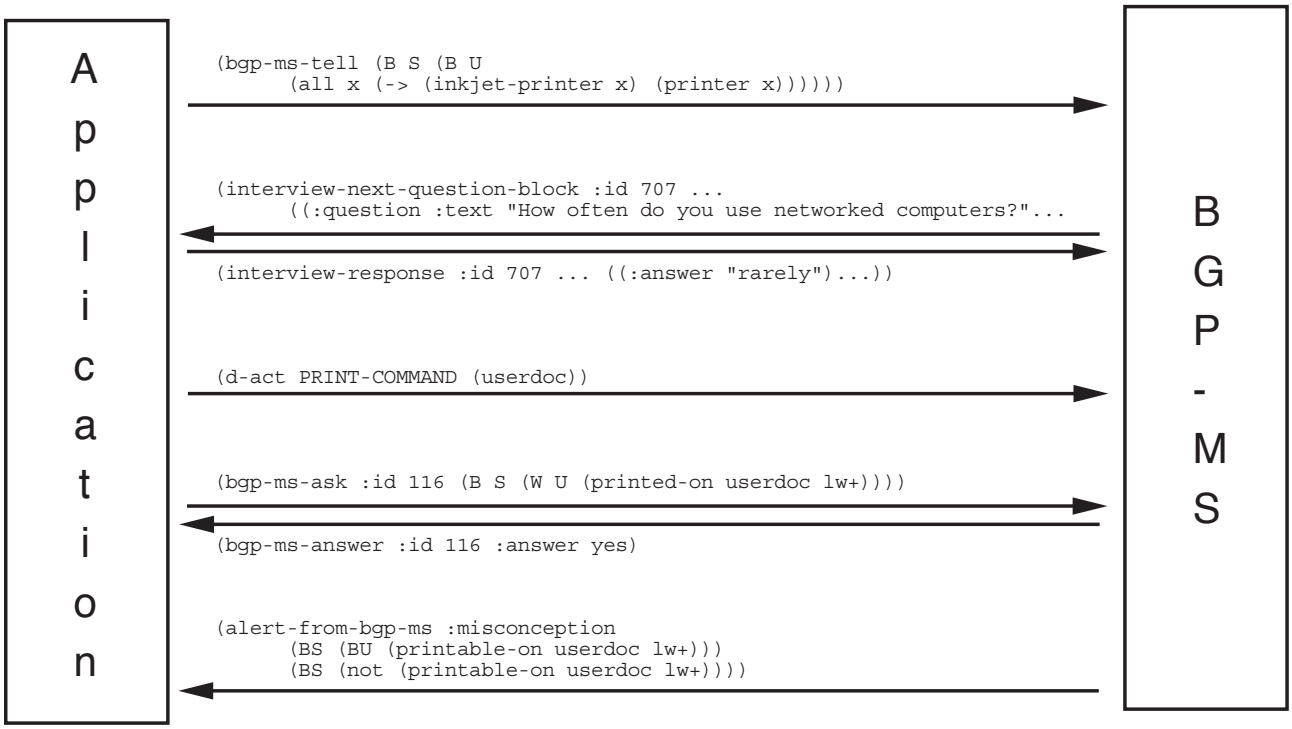

Figure 1: Interaction between the application system and BGP-MS

The application system need not have any knowledge about the internal structure of BGP-MS, except that it must be able to send appropriate messages to BGP-MS as well as interpret the messages that come from BGP-MS. It may additionally share its domain knowledge base with BGP-MS in order to avoid redundant representation, but this is neither required nor essential.

2.2 Informing BGP-MS about observed user beliefs and goals

BGP-MS accepts information from the application system regarding the user's beliefs and goals and stores it as *primary information* in its internal representations. Belief and goal descriptions that get communicated to BGP-MS have the form $\mathcal{M}p$, where $\mathcal{M}$ describes a belief/goal nesting and p describes the content of the belief or goal. The following syntax defines the belief and goal description language (BGDL) of BGP-MS. All expressions that can be formulated using this language are permissible at runtime.

```
<bel/goal-desc> ::=  (B S <belief/goal>)
<belief/goal>   ::=  p | (<modality> <agent> <belief/goal>) |
                        (<neg> (<modality> <agent> <belief/goal>))
<modality>      ::=  B | W
<agent>         ::=  S | U | M
<neg>           ::=  NOT
```

'B' and 'W' stand for the modal operators 'believe' and 'want', respectively. The agent symbol indicates whether beliefs and goals are held by the system (S), the user (U), or whether they are mutually shared (M). The content of the belief or goal, p, must be expressible by a first-order formula (which can be augmented by the ι operator for refering to specific objects for which a proposition holds true). Currently successive operators are assumed to have different agents.[3] The reason for this restriction is the lack of a clear theory of "self-reflexive" beliefs and goals, and the lack of experience regarding the need to model such beliefs and goals for user modeling purposes.

Messages of the application that communicate observed beliefs and goals to BGP-MS are labeled `bgp-ms-tell`. Here are some examples, which are mostly taken from adaptive systems that were described in the user modeling literature.

```
(bgp-ms-tell (B S (B U (all x (-> (WHALE x) (FISH x))))))
```
The user believes that whales are fish [McCoy, 1985].

```
(bgp-ms-tell (B S (B U (:concept BACTERIAL-MENINGITIS))))
```
The user knows the concept 'bacterial meningitis' [Sleeman, 1985].

[3] Technically, successive operators refering to the same agent will be collapsed into one, and the modality will be set to 'W' if any 'W' ocurred in the original sequence, and otherwise to 'B'. Sequences like `(B S (B M (B M p)))` and `(B S (B U (W U p)))`, which contain two 'M's and two 'U's in a row, will therefore become collapsed into `(B S (B M p))` and `(B S (W U p))`, respectively.

```
(bgp-ms-tell (B S (not (B U (:concept kernel)))))
```
The user does not know the concept 'kernel' [Boyle and Encarnacion, 1994].

```
(bgp-ms-tell (B S (B M (GIVES PETER MARY BOOK))))
```
It is mutually believed that Peter gives Mary a book [Kobsa, 1985].

```
(bgp-ms-tell (B S (B M (B U (B S (iota x (GIVES x MARY BOOK)))))))
```
It is mutually believed that the user believes that the system knows who gives a book to Mary [Kobsa, 1985].

```
(bgp-ms-tell (B S (B U (all x (-> (inkjet-printer x) (printer x))))))
```
The user believes that inkjet printers are printers.

```
(bgp-ms-tell (B S (B M (B U (all doc
    (-> (sent-for-printing doc) (exists p (printed-on doc p)))))))))
```
It is mutually believed that the user believes that every document that gets sent for printing will be printed on some printer.

Beliefs and goals with a leading operator 'B M' are *mutually believed* [Schiffer, 1972; Clark and Marshall, 1981; Kobsa, 1989]. If a belief or goal <belief/goal> is mutually believed, then both S and U believe that <belief/goal> is the case, believe that the other believes that <belief/goal> is the case, etc.

2.3 Interviewing the user

Initial interviews are a major source of information about the user [Rich, 1979; Peter and Rösner, 1994; Boyle and Encarnacion, 1994]. Information given by the user can be directly entered into the user model (e.g. the fact that the user is a student) and can subsequently trigger inferential processes that enter additional assumptions (e.g. by activating stereotypical assumptions about user subgroups). To date, it has always been up to the application

a) to determine the questions that must be asked,

b) to display these questions on the terminal screen and receive the user's answers,

c) to draw inferences about the user based on his answers,

d) to enter these assumptions into the user model.

BGP-MS offers an interviewing component that carries out tasks (a), (c) and (d) using rules that must be provided at the time when the application system is being developed (see Section 4.2). The application can ask the interview component for the next set of questions that should be displayed to the user, as well as the possible answers to these questions. It must display these questions (i.e., perform task (b)) and inform BGP-MS about the answers that were given by the user. The interviewing

component infers the resulting assumptions and enters them into the user model. Here is an example in which the interviewing component instructs the application to display a question block with a question concerning the user's level of network experience.

```
(interview-next-question-block :id 707 :name "Computers"
   .....
   ((:question
     :name network-experience
     :text "How often do you use networked computers?"
     :type (:select-one
              ("daily" "once per week" "rarely" "never")))
   .....
   ))
```

The application then returns the answer of the user:

```
(interview-response :id 707 ... ((:answer "rarely") ... ))
```

In addition to the interviewing component, an interview display component that carries out task (b) is currently being developed on Sun and PC platforms. Instead of programming an extra component into the application system that handles user interviews, the application developer may choose to take advantage of this component. Basic layout features of the display (e.g. the background color and the headline) have been parameterized and can be customized to the needs of the application. Fig. 2 shows the SUN display of a question block that contains the above-mentioned question.

2.4 Informing BGP-MS about observed user actions

In Section 2.2 we described how the application may communicate information about beliefs and goals of the user to BGP-MS. BGP-MS does not care how the application obtained this information. In some cases, it may have been provided directly by the user. Most times, however, the application will probably have to draw inferences based on the user's input. To relieve the application from this task, BGP-MS additionally allows the application to communicate observed user actions (more precisely: the communicative content underlying these actions), and BGP-MS itself will draw the possible assumptions about the user's beliefs and goals based on definitions of action types provided by the system developer (see Section 4.3).

Messages reporting observed user actions have the following form:

```
(d-act <action-type> (<parameters>))
```

Examples include:

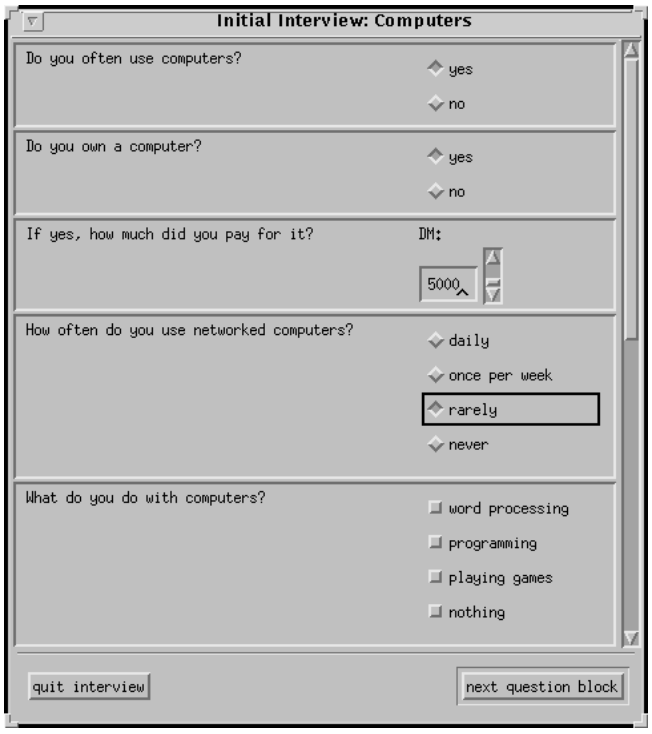

Figure 2: Example of an interview question block

`(d-act AGREE ((erase disk)))`
The user agreed that 'disk' may be erased (e.g., by clicking an OK button).

`(d-act REQUEST-EXPLANATION ((:concept kernel)))`
The user requested an explanation of the concept 'kernel' [Boyle and Encarnacion, 1994].

`(d-act PRINT-COMMAND (userdoc))`
The user executed a print command for 'userdoc'.

2.5 Asking BGP-MS for current assumptions about user beliefs and goals

The application can ask BGP-MS whether specific assumptions concerning the user's beliefs and goals are currently contained in the user model. Messages for communicating queries are labeled `bgp-ms-ask`. They contain an identifier labeled `:id` that will be used in the reply of BGP-MS. The queried assumptions are specified using the belief and goal description language described in Section 2.2. Here are some examples:

`(bgp-ms-ask :id 112 (B S (B U (:concept OTITIS-MEDIA))))`
Does the user know the concept 'otitis media'? [Sleeman, 1985]

```
(bgp-ms-ask :id 113 (B S (not (B U (:concept VRM)))))
```
Does the user not know the concept 'VRM'? [Boyle and Encarnacion, 1994]

```
(bgp-ms-ask :id 114 (B S (B U (all x (-> (equity x) (security x))))))
```
Does the user know that equities are securities? [McCoy, 1985]

```
(bgp-ms-ask :id 115 (B S (B M (GIVES PETER MARY BOOK))))
```
Is it mutually believed that Peter gives Mary a book? [Kobsa, 1985]

```
(bgp-ms-ask :id 116 (B S (W U (printed-on userdoc lw+))))
```
Does the user want to get 'userdoc' printed on the printer 'lw+'?

In order to answer a query, BGP-MS will look up the current user model to see whether it explicitly contains the assumption described in the query. For example, an answer to the last question would appear as:

```
(bgp-ms-answer :id 116 :answer yes)
```

In addition, the developer may specify that a deductive inference process (see Section 4.5) should be started if the queried description does not correspond to anything explicitly contained in the current user model. Thus, implicit information can also be taken into account. The answer to the query will be positive if the queried item is among the stored assumptions or is deducible from them.

2.6 Signaling important inferred assumptions to the application

Normally the information flow from BGP-MS to the application is driven by the demands of the application, which communicates queries to BGP-MS relating to the contents of the user model. Since assumptions about the user can also be acquired by input-driven, forward-directed inference processes (see Section 4.5.3), in exceptional cases it can be very useful for the application system to become immediately informed about certain events or constellations in the user model that may be crucial for its adaptive behavior. For instance, an application may want to get immediately informed when BGP-MS infers misconceptions or new user goals, so that it can react to this fact without delay.

Examples for messages that BGP-MS can send to inform the application about important events include the following:

```
(alert-from-bgp-ms :retracted-assumption
    (B S (B U (printable-on userdoc lw+))))
```
An assumption in the user model was retracted (e.g., by the truth maintenance component).

```
(alert-from-bgp-ms :misconception
     (B S (B U (printable-on userdoc lw+)))
     (B S (not (printable-on userdoc lw+))))
```
A misconception (i.e. a conflict between the user's and the system's beliefs) was discovered in the user model.

```
(alert-from-bgp-ms :stereotype-change
   (:just-activated NETWORK-NOVICE) (:just-retracted DOS-PROGRAMMER))
```
New stereotypes were activated and/or retracted.

```
(alert-from-bgp-ms :inferred-assumption
     (B S (W U (printed-on userdoc lw+))))
```
An assumption that matches a pre-specified pattern has been inferred in the user model.

Messages of type `alert-from-bgp-ms` merely inform the application about the occurrence of certain pre-specified events or constellations within BGP-MS. The proper handling of these events is completely left either to the application (as in the case of misconceptions) or to the automatic processing by BGP-MS (as in the case of inconsistencies and the activation or retraction of stereotypes). Communication or even negotiation between the application and BGP-MS about these events cannot take place so far, unless it remains restricted to the message types listed in Section 2.1.

2.7 Inter-process communication between the application and BGP-MS

The communication between the application and BGP-MS is being carried out via KN-IPCMS[4], a platform-independent message-oriented communication protocol that allows both for synchronous and asynchronous communication. Messages of the application containing observations about user beliefs and goals (see Section 2.2) as well as observed user actions (Section 2.4) should be transferred asynchronously. This means that the application and BGP-MS run concurrently, i.e. the user model management should largely be performed while the application is interacting with the user. Questions posed to BGP-MS concerning the current assumptions about the user (Section 2.5) or the next set of interview questions (Section 2.3) should be handled synchronously (i.e., the application will wait for the answer), and will be prioritized over incoming observations.

In order to use KN-IPCMS for communicating with BGP-MS, application developers must integrate a set of pre-defined C functions into their program environment that implement the KN-IPCMS protocol (see Section 9). [5] At runtime, the application must register as a participant of the communication process by stating its port name through which BGP-MS can communicate with it.

[4] KN-IPCMS stands for KoNstanz Inter-Process Communication Management System.

[5] For CMU Lisp, IF Prolog and ANSI-C, KN-IPCMS has already been integrated into the respective language.

3 Representing user modeling knowledge and domain knowledge

BGP-MS provides an integrated suite of knowledge representation mechanisms for representing its assumptions about the current user, its domain-specific user modeling knowledge, and optionally its general knowledge about the application domain. The outer representation layer consists of so-called *partitions,* which may be ordered in hierarchies. Within partitions, knowledge and assumptions can be represented in a conceptual representation scheme as well as in first-order predicate calculus. This two-level representation permits the representation of assumptions about the user's beliefs and goals with respect to the application domain, the system's knowledge about the application domain, and assumptions concerning application-relevant characteristics of user subgroups (the so-called *stereotypes*). Inference rules that are based on observed beliefs and goals, user responses in an interview, user actions, and the contents of the current user model can also be represented. This section will describe the representation of the first three types of knowledge, while Section 4 will deal with inference rules.

3.1 Partition hierarchies for representing individual user models and stereotypes

The partition mechanism KN-PART [Scherer, 1990; Fink and Herrmann, 1993], which forms the outer knowledge representation layer in BGP-MS, allows one to collect different types of assumptions about the user as well as the system's domain knowledge in separate partitions. A partition approach for belief and goal modeling has already been used by [Cohen, 1978; Kobsa, 1985; Ballim and Wilks, 1991] and others. In KN-PART, this approach was enhanced by subordination relations that allow for the definition of partition hierarchies with partition inheritance. If two or more partitions $P_1, \ldots, P_n$ have a common superordinate partition P_s, then KN-PART will automatically propagate the common contents of $P_1, \ldots, P_n$ to P_s (unless the application developer prohibits this in the definition of P_s), and each of the $P_1, \ldots, P_n$ will inherit the contents of P_s at the time of retrieval.

KN-PART allows the definition of arbitrary partition hierarchies (i.e., directed acyclic graphs). Leaf partitions in a hierarchy (with all their directly or indirectly inherited contents) are called *views*. Hierarchical relationships do not extend the expressive power of partitions. However, they offer a convenient means for collecting the commonalities of two or more views and, when used for the modeling of beliefs and goals, can efficiently implement certain frequently-used inferences. Views seem to fulfill Levesque's [Levesque, 1986] criteria for a vivid representation (see [Kobsa, 1992]). In BGP-MS they are employed for collecting the domain knowledge of BGP-MS, the different types of assumptions that are contained in the individual user model, and the user stereotypes that are known to BGP-MS.

3.1.1 Representation of the domain knowledge of BGP-MS

Domain knowledge which BGP-MS possibly needs for carrying out its user modeling tasks must be stored in a separate partition, which is usually named 'SB' for 'System Believes'. For applications

where BGP-MS does not need domain knowledge, SB need not be present (or may be present but empty). If SB does contain domain knowledge, the application can access it with messages of the form `(bgp-ms-tell (B S p))` and `(bgp-ms-ask (B S p))` as was described in Sections 2.2 and 2.5. Hence the application developer may choose to represent some or all domain knowledge of the application in BGP-MS rather than in the application, or duplicate some of the domain knowledge of the application into BGP-MS.

3.1.2 Representation of the individual user model

The individual user model in BGP-MS contains all current assumptions and information about the user. The types of assumptions that should be distinguished depend largely on the user modeling needs of the application system. Frequently employed types include the following:

- the "privately" held assumptions of the system about the user's beliefs about the application domain (named SBUB for System Believes User Believes), including those user beliefs which the system does not share (i.e., the user's misconceptions),
- the mutual beliefs of the system and the user about the user's beliefs about the application domain (SBMBUB), including those user beliefs which the system does not share (i.e., the mutually known user misconceptions),
- the mutual beliefs of the system and the user about the application domain (SBMB),
- the "privately" held assumptions of the system about the user's goals with respect to the application domain (named SBUW for System Believes User Wants),
- the mutual beliefs of the system and the user about the user's goals with respect to the application domain (SBMBUW),
- system assumptions about "objective facts" concerning the user (which are part of SB).

In BGP-MS, each of these assumption types is collected in a separate partition. Fig. 3 shows an example of a simple partition hierarchy that contains all the types listed above. SB and SBMBUB inherit from partition SBMB since all mutually shared domain knowledge is known by the system and mutually believed to be believed by the user. Mutually believed beliefs of the user, on the other hand, are believed by the user, and believed by the user to be believed by the system to be believed by the user, etc. Hence SBUB, SBUBSBUB, etc. form subpartitions of SBMBUB. The user model developer may select the types of partitions and the nesting of assumptions that are appropriate in the application domain. Experience with natural-language dialog systems give rise to the assumption that the maximum depth of embedding is quite limited in cooperative dialog situations [Taylor and Carletta, 1994].

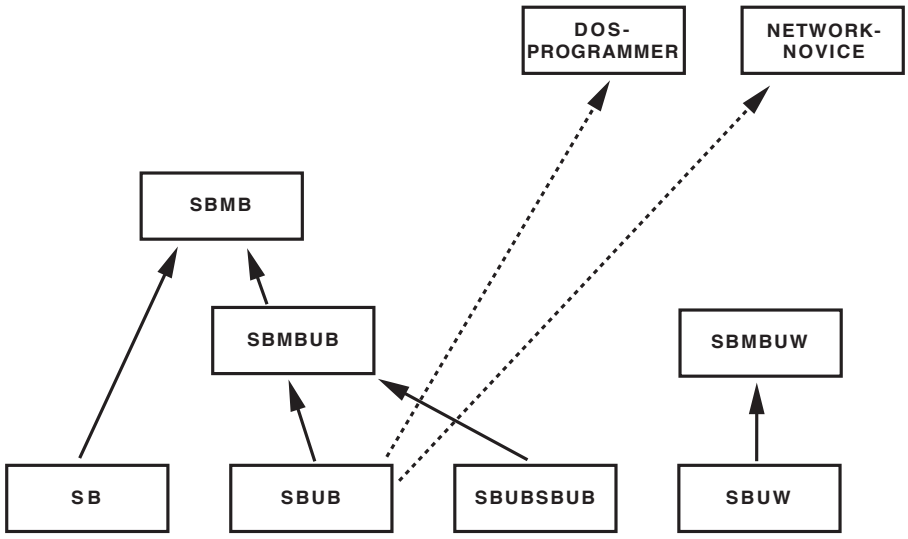

Figure 3: Domain knowledge, individual user model and stereotypes represented in a partition hierarchy

3.1.3 Representation of stereotypes

In the stereotype approach to user model acquisition, the application developer must identify subgroups within the expected user population whose members are very likely to possess certain homogeneous application-relevant characteristics. The characteristics of the identified user groups must be formalized in an appropriate representation system. The collection of all represented characteristics of a user subgroup is called a stereotype for this subgroup. For each stereotype, the application developer must also identify a small number of key characteristics which allow a user modeling system to identify the user as belonging to the corresponding user subgroup, or as not belonging to this subgroup. Knowledge about these key characteristics is encoded in so-called activation and retraction conditions for each stereotype.

In BGP-MS, each stereotype is represented by a separate partition. If it applies to the current user, it is linked by an inheritance link to an appropriate partition of the individual user model, which thereby automatically inherits the contents of the stereotype. Fig. 3 contains two stereotype partitions that collect the presumable domain knowledge of DOS programmers and network novices. They can become linked to partition SBUB if their activation conditions have been met (this is symbolized by the dotted lines).

If the contents of two or more stereotypes overlap, partition hierarchies may optionally be used for constructing stereotype hierarchies. Workspace may thereby be saved since the contents of superordinate stereotype partitions become inherited by the subordinate partitions and hence need only be represented once. In [Kobsa, 1990], several types of stereotype hierarchies were discussed that have so far been employed in the user modeling literature (but were not very much substantiated empirically). While partition hierarchies in BGP-MS may be arbitrary directed acyclic graphs,

partitions must fulfill two minor additional restrictions in order that they can be declared as stereotype partitions. One condition is that stereotype partitions must be leaves of the stereotype hierarchy, or possess at least two children, all of which must be stereotype partitions. The other condition is that upward propagation from non-stereotype partitions to stereotype partitions is prohibited. The purpose of both restrictions is to avoid incorrect upward propagation.

3.2 Hybrid representation within partitions

As we have shown in the previous section, partitions may serve as storages for the different types of assumptions about the user, allowing the developer to separate different parts of the user model. Within partitions, assumptions can be represented in a hybrid formalism that consists of two components. The conceptual knowledge representation language SB-ONE [Kobsa, 1990; Kobsa, 1991] can be used to formulate assumptions concerning the concepts of the application domain. The language of first-order predicate calculus is employed for expressing assumptions that cannot be represented in SB-ONE, notably expressions that include disjunction, negation, quantification and implication.

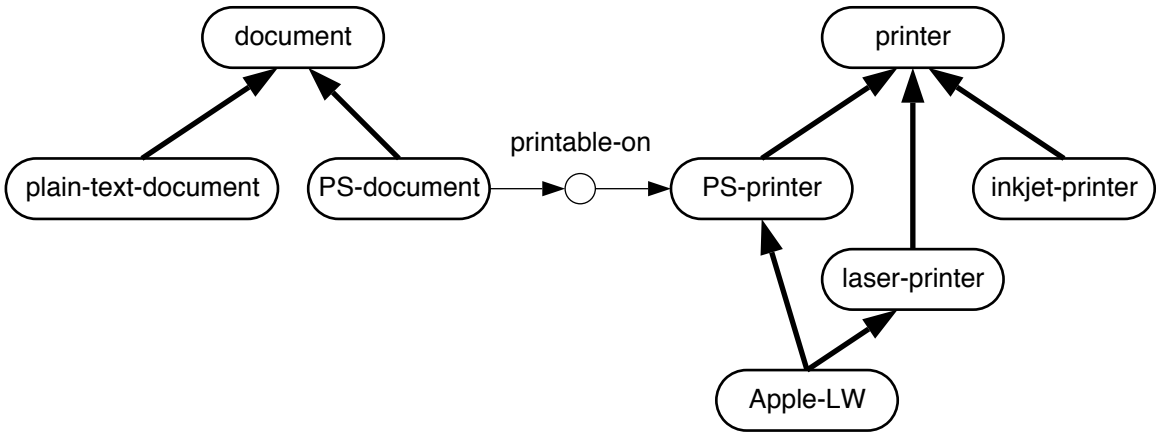

Figure 4: Domain knowledge about concepts involved in printing documents (represented in SB-ONE)

SB-ONE is a representation language for conceptual knowledge which fits loosely into the KL-ONE paradigm [Brachman, 1978; Brachman and Schmolze, 1985]. A detailed discussion of SB-ONE is beyond the scope of this paper. We will only give an example to demonstrate its use in user modeling systems. We thereby assume that the reader is roughly familiar with KL-ONE-like formalisms. Fig. 4 shows a possible SB-ONE concept hierarchy that represents knowledge about printing documents. Assume that this is part of the domain knowledge of an application; hence it is contained in partition SB. The system (the application) knows documents and printers. Additionally, it knows that PostScript (PS) printers, laser printers and inkjet printers are printers, and that Apple laser writers are both PS printers and laser printers. Moreover, it knows about two different kinds of documents: PS documents, which can be printed on PS printers, and plain text documents. In SB-ONE, all this can be expressed using general concepts (‘printer’, ‘document’, ‘PS-printer’, ‘laser-printer’, ‘inkjet-printer’, ‘Apple-LW’, ‘PS-document’, and ‘plain-text-document’), subsumption relations (‘printer’ subsumes

'PS-printer', 'laser-printer', and 'inkjet-printer'; 'Apple-LW' is subsumed by both 'PS-printer' and 'laser-printer'; and 'document' subsumes 'PS-document' and 'plain-text-document') and the role 'printable-on', which is attached to 'PS-document' and value-restricted to 'PS-printer'.

The second means for representing assumptions within partitions is first-order predicate calculus (FOPC). It was introduced into BGP-MS to allow more assumptions to be made that cannot be expressed in SB-ONE. Examples include the user's rule-like knowledge about the application domain. For instance, in our printing domain the user may be assumed to believe in the rule:

Every document that is printed is printed on the user's office printer.

This rule-like belief of the user can be formalized by the following FOPC expression:

$$\forall doc, p \ \text{[printed}(doc) \wedge \text{office-printer(*current-user*}, p) \rightarrow \ \text{printed-on}(doc, p)]$$

In addition to such simple rules, also negative, disjunctive and quantified assumptions – isolated or within complex rules – can be expressed in FOPC (SB-ONE permits such expressions only in very restricted forms). For instance, a user may believe that 'lw+' is a laser printer or inkjet printer, and that plain text documents won't print on PS printers:

$$\text{laser-printer(lw+)} \vee \text{inkjet-printer(lw+)}$$

$$\forall doc, p \ \text{[plain-text-document}(doc) \wedge \text{PS-printer}(p) \rightarrow \ \neg\text{printable-on}(doc, p)]$$

The expressive power of FOPC exceeds that of SB-ONE. Hence every SB-ONE structure is in principle expressible in FOPC. From the perspective of expressive power alone, it would not be necessary to include SB-ONE in the set of representation formalisms of BGP-MS. However, there are two main advantages of a hybrid representation:

1. SB-ONE is a specialized formalism for the representation of taxonomies of object classes and binary relations between these classes. Quite a number of applications will probably only need to represent such terminological knowledge. (As a matter of fact, this already holds true for the first application that employs BGP-MS – see Section 8.) If such applications had to use FOPC instead of SB-ONE, unnecessary representational overhead and inefficiency would be forced upon them.
2. Many inferences that can be drawn on the basis of knowledge that is representable in SB-ONE are already "hard-wired" in the SB-ONE knowledge representation system. Hence the terminological part of the modeled user knowledge should be better represented with SB-ONE, even when FOPC is needed in addition to SB-ONE.

It is particularly for the latter reason that reported beliefs and goals should be preferably represented in SB-ONE. The component of BGP-MS that translates expressions in the belief and goal description language into the knowledge representation formalisms of BGP-MS (see Section 4.1) implements this strategy. Integration mechanisms (see Section 4.5.1) take care that SB-ONE encoded knowledge is taken into account in FOPC inferences.

4 Drawing inferences about the user

BGP-MS provides several inference methods for deriving assumptions about the user based on messages from the application and previously made assumptions. It is possible that one message triggers more than one inference method. In addition, application developers may define Lisp functions that BGP-MS will execute upon the receipt of `bgp-ms-tell` messages. Arbitrary domain-specific inference processes may thereby be added to those that have already been implemented.

4.1 Inferences based on observed beliefs and goals

Observed user beliefs and goals that the application reports to BGP-MS using the belief and goal description language presented in Section 2.2 become fairly directly entered into the current user model. The fact that the information was communicated by the application becomes recorded as well. In the future, the truth-maintenance component of BGP-MS will withdraw inappropriate assumptions if inconsistencies between new and existing assumptions are detected (possibly after inferences were performed).

The mapping of expressions of the belief and goal description language (BGDL) to entries for the current user model is carried out by the BGP-MS function 'bgp-ms-tell'. Two cases must be distinguished, depending on whether the BGDL expression under consideration is 'modal-negated' (i.e., contains at least one modal operator that is preceded by a negation symbol), or whether this is not the case. The first case will be discussed later in Section 4.5.2. In the second case, the mapping is rather straightforward, as far as the modal part is concerned. There exists a one-to-one correspondence between a modal sequence $\mathcal{M}$ and a partition name; only the symbols that denote the modality and the agent have to be switched. For instance, when BGP-MS receives a message `(bgp-ms-tell (B S (B M (W U` *p*`))))`, it will enter the first-order expression *p* into the partition SBMBUW.

If the application developer decides to include both SB-ONE and FOPC into the knowledge representation of his user modeling system, the problem arises in which formalism *p* should be expressed. The knowledge categorization component KN-TRANS [Winkler, 1994] has been developed which assigns *p* to the appropriate formalism. If SB-ONE is included and *p* expressible in SB-ONE, *p* will become translated into SB-ONE structures and entered into the appropriate partition. If *p* is not expressible in SB-ONE, it will be entered unchanged as a first-order expression. Hence the belief description in the message

```
(bgp-ms-tell (B S (B M (B U
  (all x (-> (inkjet-printer x) (printer x)))))))
```

will become translated into an SB-ONE subsumption relation between the concepts 'printer' and 'inkjet-printer' in partition SBMBUB. In contrast,

```
(bgp-ms-tell (B S (B M (B U
  (all doc (-> (sent-for-printing doc)
               (exists p (printed-on doc p))))))))
```

will result in '$\forall doc$ [sent-for-printing(doc) $\rightarrow$ $\exists p$ printed-on(doc, p)]' being entered in partition SBMBUB.

The algorithm of the knowledge categorization component gives priority to expressing assumptions about the user in SB-ONE, if possible. As a prerequisite for the translation into SB-ONE, the component contains descriptions of all SB-ONE representation constructs in the form of first-order formula patterns. These patterns are compared with the expression *p* and, if no direct match is found, with automatically generated expressions that are logically equivalent to *p*. If a match is found, *p* can be represented using the SB-ONE construct associated with the matching pattern.

4.2 Inferences from user interviews

In section 2.3 we described the communication between the application and the interviewing component. Here we will explain how the application developer can define the interview, and how inferences about the user are drawn from his answers based on the interview definition.

An interview consists 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 are sent back to BGP-MS. The user may choose to skip individual questions, question blocks, or even the whole interview.

The definition of a question within a question block includes its 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 pre-defined 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 the BGP-MS belief and goal description language.

Fig. 5 shows the definition of an interview question by which the user's degree of network experience is to be determined. The possible answers form a set containing "daily", "once per week", "rarely", and "never". If the answer is "rarely" or "never", the expression 'network-novice(*current-user*)' will be entered into partition SB. If the answer is "daily", the expression 'network-experienced(*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, and become entered into the user model. The interviewing component can take these assumptions into account both when it composes the next question block and when it draws the conclusions from new answers. BGP-MS therefore allows for conditional expressions in the definition of the interview sequence and the conclusion parts of question definitions.

```
(define-interview-question
  :name network-experience
  :text "How often do you use networked computers?"
  :type (:select-one ("daily" "once per week" "rarely" "never"))
  :conclusions
    (((member answer '("rarely" "never"))
      (B S (network-novice *current-user*)))
     ((equal answer "daily")
      (B S (network-experienced *current-user*)))))
```

Figure 5: Definition of an interview question

4.3 Inferences based on the user's actions

For acquiring assumptions about the user, acquisition heuristics have frequently been employed in the user modeling literature. These rules 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 carries out specific actions at the user interface [Pohl *et al.*, 1995]. The assumptions that are made with many of these heuristics can be regarded as prerequisites to the felicitous execution of the respective actions. For example, the correct use of an object presumes that the user knows the object; hence he can be assumed to know the object if he 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 he can be assumed to know the object if he requests technical details for it [Boyle and Encarnacion, 1994].

This reminds one of the presupposition analysis method that has been applied for supporting the acquisition of dialog partner models in natural-language dialog systems [Kobsa, 1985]: A user utterance is analyzed with respect to the speech acts [Searle, 1969] it verbalizes, and the so-called presuppositions, which must have been valid for the speaker to perform the acts correctly, are derived from each speech act. 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 BGP-MS, we generalized the notion of natural-language speech acts to *dialog acts* that may occur in human-computer interaction via any kind of user interface, following other speech-act based approaches in this area [Winograd, 1988; Sitter and Stein, 1992]. The assumptions about the user that can be drawn from these dialog acts are interpreted as their presuppositions. Like speech acts, dialog acts can be categorized under *dialog act types*, which are normally 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, *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 interactive

computer systems. BGP-MS therefore offers the application developer not only a library with general dialog act types (like INFORM, AGREE, and two general kinds of questions), but also the possibility to define new dialog act types that may serve as acquisition rules for an application.

```
(define-d-act :name AGREE :parameters (topic)
              :presupp ((B S (B M (not (W U (not topic)))))))

(define-d-act :name REQUEST-EXPLANATION :parameters (topic)
              :presupp ((B S (B M (not (B U topic))))))

(define-d-act :name PRINT-COMMAND :parameters (document)
              :presupp ((B S (B M (sent-for-printing document)))
                        (B S (B M (W U (printed document))))))
```

Figure 6: Definitions of dialog act types

Fig. 6 shows the definitions for the dialog act types used in the examples of Section 2.4. If an instance of the AGREE dialog act type is observed by the application and communicated to BGP-MS, BGP-MS will infer that henceforth it is mutually believed that the user does not want the negation of its parameter 'topic'. Hence, the message of the application

```
(d-act AGREE ((erase disk)))
```

will result in the following call to the function bgp-ms-tell:

```
(bgp-ms-tell '(B S (B M (not (W U (not (erase disk)))))))
```

The REQUEST-EXPLANATION type also has one parameter, which is the topic to be explained by the application. If an instance of it occurs, it will henceforth be assumed that the user does not know this topic. The message

```
(d-act REQUEST-EXPLANATION ((:concept kernel)))
```

therefore results in:

```
(bgp-ms-tell '(B S (B M (not (B U (:concept kernel))))))
```

An instance of the dialog act type PRINT-COMMAND occurs when the user tries to print a document, e.g. with 'lpr' in UNIX or by selecting the 'print' item in a menu-based interface. Its parameter is the document to be printed. When it occurs it will be inferred that system and user mutually believe that the document is being sent to a printer and that the user is from now on mutually believed to have the goal that the document is printed. In our example, the action

```
(d-act PRINT-COMMAND (userdoc))
```

is observed, which yields the following two function calls:

```
(bgp-ms-tell '(B S (B M (sent-for-printing userdoc))))

(bgp-ms-tell '(B S (B M (W U (printed userdoc)))))
```

The expression 'sent-for-printing(userdoc)' will therefore become entered in partition SBMB and 'printed(userdoc)' in SBMBUW.

4.4 Activation and retraction of stereotypes

BGP-MS offers a facility that automatically activates the stereotypes that apply to the current user and retracts active stereotypes that later turn out to be inappropriate. If the application developer decides to take advantage of this facility, he has to define the activation and retraction conditions for each stereotype. Both refer to the current user model, but only take the most directly obtained (and hence probably most reliable) assumptions about the user into account. These include beliefs and goals that were reported by the application, as well as assumptions made on the basis of the user's answers in the interview and the dialog acts he performed. Assumptions acquired by deductive inference processes or through the activation of other stereotypes are not considered.

BGP-MS offers a set of pre-defined condition schemes which the application developer can appropriately instantiate to define the activation and retraction conditions of the stereotypes in his application domain. Frequently used schemes include

- IFKNOWN *list*: is satisfied if all knowledge items contained in *list* are known to the user (i.e., are all contained in view SBUB).
- IFUNKNOWN *list*: is satisfied if the knowledge items contained in *list* are all unknown to the user.
- IFKNOWN% *n*: is satisfied if at least *n* percent of the contents of the stereotype are known to the user.
- IFKNOWN%OF *n list*: is satisfied if at least *n* percent of the knowledge items contained in *list* are known to the user.

The application developer may define additional condition schemes in Lisp. For defining the activation and retraction conditions of stereotypes, the schemes must be instantiated with appropriate knowledge items of the application domain. Instantiated schemes can be logically combined by the connectives AND, OR, and NOT, and also with any Lisp code that returns a Boolean value. The definition of

```
(or (bgp-ms-ask '(B S (network-novice *current-user*)))
    (IFUNKNOWN '((:concept LAN) (:concept ethernet)
                 (:concept file-server))))
```

Figure 7: Activation condition of the stereotype NETWORK-NOVICE

stereotype activation and retraction conditions and of supplementary condition schemes is facilitated by an easy-to-use editor, which is part of the graphical interface of BGP-MS (see Section 6).

Fig. 7 shows an example of a stereotype activation condition that is associated with the NETWORK-NOVICE stereotype (see Section 3.1.3). It will be satisfied if the fact 'network-novice (*current-user*)' is contained in view SB[6], or if none of the concepts 'LAN', 'ethernet' and 'file-server' are contained in view SBUB. If this is the case, the partition SBUB will be connected to the stereotype partition NETWORK-NOVICE, and will automatically inherit its contents. The application developer can determine how many new assumptions concerning the user must be communicated by the application, the interviewing component or the dialog act analysis component before the stereotype activation and retraction conditions become re-evaluated.

4.5 Inferences within the individual user model

4.5.1 Inferences within Views

The representation formalisms available in BGP-MS offer a variety of inference possibilities, which are carried out by two main inference processes. The representation system for conceptual knowledge, SB-ONE, offers numerous built-in inferences. The resolution-based theorem prover OTTER [McCune, 1994] is employed for inferences in FOPC. Since the expressive power of SB-ONE is considerably weaker than that of first-order predicate calculus (see Section 3.2), OTTER was chosen as the central processor for inferences that take place within views. OTTER consults SB-ONE with regard to conceptual knowledge, if this seems promising.

Examples for built-in SB-ONE inferences include the automatic computation of the transitive closure of the subsumption relation: if the system knows that Apple laser writers are PostScript printers, and that PostScript printers are printers, it also knows that Apple laser writers are printers. Another built-in inference process is classification, which positions a concept in the concept hierarchy based on its description by roles and value restrictions for role fillers.

Inferences that are usually carried out on the basis of FOPC include, e.g., the execution of inference rules which the user presumably employs. Such inference rules normally cannot be represented in SB-ONE since (unlike FOPC) SB-ONE does not allow general implication. These inference rules are ascribed to the user and form part of his individual user model. Hence they need not be valid, but may be misconceptions, and may be used to simulate the (potentially fallacious) reasoning of the user.

[6] To determine this, the BGP-MS function 'bgp-ms-ask' is used, which processes the namesake messages.

Logical inferences within a view are performed by OTTER via a resolution process. If a formula logically follows from a set of other formulas, OTTER will be able to show this, and vice versa (given unlimited processing time). For example, assume that the NETWORK-NOVICE stereotype partition contains the rule

$$\forall doc, p \ [\text{printed}(doc) \wedge \text{office-printer}(\text{*current-user*}, p) \rightarrow \ \text{printed-on}(doc, p)]$$

After NETWORK-NOVICE has been activated, this rule is also visible in SBUB (see Fig. 3). Assume further that the facts 'printed(userdoc)' and 'office-printer(*current-user*,lw+)' are contained in partition SBMB, and are hence also visible in view SBUB. Then the above rule can be used to infer in view SBUB that the user believes that 'userdoc' will be printed on printer 'lw+', which is represented by the derived fact 'printed-on(userdoc,lw+)'.

Since the division of view contents into logical and conceptual knowledge is hidden from the application through the automatic categorization of expressions of the belief and goal description language, it must be guaranteed that the communicated assumptions are all considered by the inference component, no matter whether they are represented in FOPC or SB-ONE. Logical and conceptual knowledge must therefore be integrated within views. Since OTTER was selected as the central inference engine, the conceptual knowledge has to be integrated into the logical inference process. For this purpose, OTTER has been enhanced by a component that recognizes situations in the resolution process in which a query to the SB-ONE part of a view may be beneficial [Zimmermann, 1994].

Here is an example of an inference that considers both logical and conceptual knowledge. Assume that the conceptual hierarchy of Fig. 4 is contained in view SB. Additionally, the system knows the rule that plain text documents cannot be printed on PostScript printers (cf. Section 3.2):

$$\forall doc, p \ [\text{plain-text-document}(doc) \wedge \text{PS-printer}(p) \rightarrow \ \neg\text{printable-on}(doc, p)]$$

Moreover, assume that SBMB contains the mutual beliefs 'plain-text-document(userdoc)' and 'Apple-LW(lw+)'. Since 'Apple-LW' is subsumed by 'PS-printer', SBMB also implicitly contains 'PS-printer(lw+)'. Since SB inherits from SBMB, all these facts are also contained in SB. Hence BGP-MS can infer in SB that

$$\neg\text{printable-on}(\text{userdoc}, \text{lw+})$$

4.5.2 Inferences across views

Modal logic for view-connecting knowledge and modal-negated BGDL expressions Using the facilities described so far, BGP-MS can draw inferences based on all expressions of the belief and goal description language (except those that contain negated modal operators). When these expressions become communicated to BGP-MS, FOPC formulas or SB-ONE structures become entered into those partitions of the partition hierarchy that correspond to the leading belief and goal operator sequence $\mathcal{M}$. Inferences within views can then take place to derive implicit assumptions.

Although in many cases these mechanisms will satisfy the inferential needs of a user modeling system, we wanted to additionally enable the application developer to formulate statements expressing

relationships between *different* views. Relationships may exist between specific contents of views ("If the user believes A, then he will want B"), and between views in general ("Everything that is believed by the system is believed by the user"). Here we will describe how both types of view-connecting knowledge can be represented and processed in BGP-MS. Also the treatment of modal-negated BGDL expressions will be discussed.

Both view-connecting statements and modal-negated expressions can be represented in a fairly straightforward way: The belief and goal description language, which comprises all assumptions about the user that can be represented within BGP-MS, corresponds to a subset of a modal logic with indexed modal operators for beliefs and goals. Within this subset, modal-negated expressions can be represented. The following fomula expresses that the user belief 'printed-on(userdoc,lw+)' is not mutually believed:

$$B_S \neg B_M \text{ printed-on}(\text{userdoc}, \text{lw+}) \tag{1}$$

The complete version of the modal logic, which includes the standard logical connectives, can be used to express relationships between specific contents of different views. For example,

$$\forall doc, p \, [B_S W_U \text{ printed-on}(doc, p) \rightarrow B_S B_U \text{ printable-on}(doc, p)] \tag{2}$$

represents a rule for inferring an assumption about a user belief from an assumption about a user goal: if the user is believed to want a document to be printed on some printer, then he is also assumed to believe that this is possible.

General relationships between views may be expressed by modal formula schemes, which (for our purposes) result from replacing FOPC parts in modal expressions by formula variables. For example,

$$(B_S W_U \, \Phi \wedge B_S B_U \, (\Phi \rightarrow \Psi)) \rightarrow B_S W_U \, \Psi \tag{3}$$

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

Inferences in modal logic Modal logic inferences can be performed using a first-order logic reasoner. To achieve this, a procedure is needed that translates modal expressions into predicate calculus while preserving their original semantics. The so-called *functional translation* [Ohlbach, 1991] is a method based on the standard possible-worlds semantics for modal logic that transforms sequences of modal operators into complex logical terms. The transformation uses *context access functions* that map worlds to accessible worlds, and consecutively applies one function for each modal operator to the initial world. Thereby the generated terms represent paths through the possible-worlds structure. The operator sequences are then replaced by adding these "world paths" as additional arguments to all atoms in the formula. (In more complicated versions, terms have to be translated as well).

To illustrate this approach, we will first show the translation of two view-internal expressions (which are not translated in BGP-MS, but rather assigned to the appropriate partition – see Section 4.1):

$$B_S W_U \text{ printed}(\text{userdoc})$$

$$B_S B_U \, \forall doc \, [\text{printed}(doc) \rightarrow \text{printed-on}(doc, \text{lw+})]$$

are translated into [7]

$$\forall f, g \text{ printed}(g_{W_U}(f_{B_S}(\mathrm{w}_0)), \text{userdoc}) \tag{4}$$

$$\forall f, g, doc\ [\text{printed}(g_{B_U}(f_{B_S}(\mathrm{w}_0)), doc) \rightarrow \text{printed-on}(g_{B_U}(f_{B_S}(\mathrm{w}_0)), doc, \text{lw+})] \tag{5}$$

where f, g, h denote context access functions and w_0 denotes the initial world. Negations within the modal operator sequence are simply retained, so that the modal-negated formula (1) translates to

$$\forall f\ \neg\forall g \text{ printed-on}(g_{B_M}(f_{B_S}(\mathrm{w}_0)), \text{userdoc}, \text{lw+}) \tag{6}$$

The following formula is the translation of the 'view-connecting' formula (2). In contrast to (5), the two world paths in this formula are different.

$$\forall f, g, t, u\ \forall doc, p \\ [\text{printed}(g_{W_U}(f_{B_S}(\mathrm{w}_0)), doc, p) \rightarrow \text{printable-on}(u_{B_U}(t_{B_S}(\mathrm{w}_0)), doc, p)] \tag{7}$$

The functional translation even helps in integrating modal schemes into the inference procedure. Modal schemes are second order by nature because they contain implicitly quantified formula variables. First-order reasoners therefore cannot be applied. However, there is a method to avoid this problem. The SCAN algorithm [Gabbay and Ohlbach, 1992] eliminates second-order quantifiers and transforms modal schemes into a set of first-order equalities employing functional translation. These equalities between world paths represent the relations between views that were originally expressed by the modal scheme. They can be processed by OTTER's built-in equation handling.

For example, from the modal scheme (3) which relates user beliefs and goals, the following conjunction of two equations would be generated:

$$\forall w\ \forall q, r\ \exists f, g, t, u \\ [g_{W_U}(f_{B_S}(w)) = u_{B_U}(t_{B_S}(w)) \wedge \quad u_{B_U}(t_{B_S}(w)) = r_{W_U}(q_{B_S}(w))] \tag{8}$$

In these equations, the initial world constant w_0 is replaced by the universally quantified variable w. 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.

From fact (4), rule (5) and the equations (8), a predicate logical reasoner with equation handling can now infer that the user is assumed to want that 'userdoc' is printed on 'lw+' (because he wants 'userdoc' to be printed and believes that everything which is printed is printed on 'lw+'):

$$\forall f, g \text{ printed-on}(g_{W_U}(f_{B_S}(\mathrm{w}_0)), \text{userdoc}, \text{lw+}) \tag{9}$$

Taking (7) – the translation of the view-connecting rule (2) – into account, we can further infer that the user believes that 'userdoc' can be printed on 'lw+':

$$\forall f, g \text{ printable-on}(g_{B_U}(f_{B_S}(\mathrm{w}_0)), \text{userdoc}, \text{lw+}) \tag{10}$$

which is the translation of

$$B_S B_U \text{ printable-on}(\text{userdoc}, \text{lw+}) \tag{11}$$

This user belief can be regarded as a misconception, since it is contradictory to the assumption we obtained for view SB in the example for hybrid inferences within views (see the end of Section 4.5.1). BGP-MS could automatically alert the application about this situation (see Section 2.6).

[7] The quantification over function symbols seems to make these translation results second order. This is not the case, because the domain of these functions (the set of possible worlds) differs from the standard domain of the formula.

Integrating modal inferences The use of the described techniques for transforming modal expressions and schemes into predicate calculus has a severe consequence: the systematic application of these techniques renders the partition mechanism superfluous, because everything with a modality must be translated in order that it can be considered in the inference process. Consequently, even expressions within views (which correspond to non-negated modal expressions within the scope of BGDL) are affected. From a representational point of view such a translation would be possible, since our modal logic is a superset of BGDL, which again subsumes FOPC in partitions ('$\text{FOPC}_{\text{PART}}$' for short) and consequently also SB-ONE structures in partitions ('$\text{SB-ONE}_{\text{PART}}$'):

$$\text{modal logic} \supset \text{BGDL} \supset \text{FOPC}_{\text{PART}} \supset \text{SB-ONE}_{\text{PART}}$$

However, in Section 3.2 we already endorsed the preservation of SB-ONE as a view-internal representation formalism. Therefore partitions are also still necessary, at least for the representation of SB-ONE in different views. The partition mechanism is moreover advantageous in cases where inferences are constrained to a single view, since the following relationship exists between the different representation formalisms of BGP-MS (let $\prec$ mean 'is inferentially less efficient than'):

$$\text{modal logic} \prec \text{FOPC}_{\text{PART}} \prec \text{SB-ONE}_{\text{PART}}$$

Currently work is being carried out in the BGP-MS project which aims at reconciling the translation mechanisms with the partition paradigm. Our approach is based on the observation that basically world paths that are associated with literals refer to partitions. This could be used to link literals of translated view-connecting formulas to literals contained in views, in order to make integrated inference steps possible.

4.5.3 Bidirectional inferences

Inference processes within the individual user model can be executed in two different ways:

1. When answering queries of the application, BGP-MS can employ an integrated inference mechanism for backward reasoning, which verifies whether the queried item can be deduced from the current contents of the individual user model, thereby taking view-internal formulas and SB-ONE structures as well as view-connecting knowledge into account. Backward reasoning makes use of the refutation procedure implemented in OTTER: the negation of the query is formed, and a conflict between the negated query and the relevant user model contents is sought. If a conflict occurs, the queried item is explicitly contained in the user model or derivable from explicit assumptions in the user model. Inferred implicit assumptions become added to the user model in order to avoid the same reasoning steps in the future. The depth of the backward reasoning process is limited in order to restrain the number of inferred assumptions.

2. In addition, every input into BGP-MS can be examined to determine whether it leads to new implicit assumptions about the user that can be derived in a forward-directed manner. In order to accomplish this, the BGP-MS inference engine can be started in a forward-reasoning mode. The inference depth of this process is also limited in order to avoid inadequate or irrelevant derivations with a high degree of implicitness. The results are likewise added to the current user model to possibly answer future questions of the application faster by avoiding question-driven backward reasoning. This forward-directed pre-computation of inferences is even inexpensive since it can be performed concurrently with the application (whereby the idle time of the application can be used).

It should be noted that other inference mechanisms of BGP-MS are also forward-directed: the user interview component infers assumptions from the answers given by the user, as specified in the question definitions; the dialogue act analysis component infers assumptions from observed user actions depending on the actual parameters of the action; and the interpreter of the stereotype activation rules infers the contents of a stereotype partition by attaching it to the individual user model.

4.6 Inferential dependencies and consistency maintenance

Assumptions in the individual user model may possibly become contradicted by newer assumptions, no matter whether they were reported by the application or inferred by BGP-MS. In such cases it may be necessary to retract an assumption in favor of some other assumption. An additional problem arises when inferences were previously made using the assumption that is to be retracted, since then all depending assumptions possibly have to be retracted as well. In order to cope with this problem, several user modeling components and shell systems (see Section 7) employ reason-maintenance components that store and maintain the inferential dependencies between the assumptions in the user model, and keep track as to whether an assumption stems from an observation of the application, and whether it is currently justified by an inference. If an assumption becomes retracted it is not literally removed. Instead its status is changed, the justification of dependent assumptions is recalculated, and as a consequence some or all dependent assumptions possibly become retracted as well.

Currently, a reason maintenance component for BGP-MS is being developed. When a new assumption about the user is made that is contradictory to an already existing assumption, strategies for resolving this conflict must exist. Such strategies will include the assignment of priorities to assumptions based on how and when they were derived. In contrast to other systems with a reason maintenance component, both the knowledge representation and the inference processes of BGP-MS are much richer and more complicated, which makes the reason-maintenance task more difficult. For example, inferences across views must be dealt with, and SB-ONE structures must be integrated in a justification if the concerned assumption was inferred by hybrid reasoning. Moreover, the retraction of stereotypes should also be included into the reason-maintenance algorithm, which so far is not the case for any user modeling shell system.

5 Example of the operation within BGP-MS

In this section we will present an example in which almost all BGP-MS components will be used. Imagine the following scenario: A user of a networked computer is categorized as a network novice due to his answer to an interview question. As a member of this category, he is supposed to believe that every issued print-job will be printed on the PostScript printer in his office (he is not aware of the possibility that network software may have other default printers). Later he is observed to invoke a print-command on a plain text document. By hybrid and view-connecting forward inferences, it can be derived that the user believes his document is printable on his office printer, but that the system believes just the opposite. This conflict is then signaled to the application.

The following summarizes the definitions and declarations made by the application developer that are relevant to this example. All of them have already been presented at different places in previous sections.

1. prerequisite: the partition hierarchy of Figure 3;
2. the definition of an interview question:

```
(define-interview-question
  :name network-experience
  :text "How often do you use networked computers?"
  :type (:select-one ("daily" "once per week" "rarely" "never"))
  :conclusions
    (((member answer '("rarely" "never"))
      (B S (network-novice *current-user*)))
     ((equal answer "daily")
      (B S (network-experienced *current-user*))))
  )
```

3. the activation rule for the NETWORK-NOVICE stereotype and the simplistic printing rule contained in this stereotype:

```
(or (bgp-ms-ask '(B S (network-novice *current-user*)))
    (IFUNKNOWN '((:concept LAN) (:concept ethernet)
                 (:concept file-server))))
```

$\forall doc, p$ [printed(doc) $\wedge$ office-printer(*current-user*, p) $\rightarrow$ printed-on(doc, p)]

4. part of the system's knowledge about printing documents, as contained in partition SB; namely, a conceptual hierarchy concerning printers and documents (see Fig. 4) as well as the logical rule

$\forall doc, p$ [plain-text-document(doc) $\wedge$ PS-printer(p) $\rightarrow$ $\neg$printable-on(doc, p)]

5. two facts which the application developer has entered into partition SBMB: 'Apple-LW(lw+)' and 'office-printer(*current-user*,lw+)';

6. the definition of the 'print-command' dialogue act:

```
(define-d-act :name PRINT-COMMAND :parameters (document)
              :presupp ((B S (B M (sent-for-printing document)))
                        (B S (B M (W U (printed document))))))
```

7. two rules representing the system's knowledge about relationships between the views SBUB and SBUW (where the first rule is context-specific and the second one general):

   $\forall doc, p\ [B_S W_U\ \text{printed-on}(doc, p) \rightarrow\ B_S B_U\ \text{printable-on}(doc, p)]$

   $(B_S W_U\ \Phi \wedge B_S B_U\ (\Phi \rightarrow \Psi)) \rightarrow B_S W_U\ \Psi$

8. the declaration that misconceptions of the user (i.e., SBUB entries that are contradictory to entries in partition SB) should be signaled to the application.

Now we will have a look at what happens during runtime. Employing the interview question named 'network-experience' (cf. item (2)) the application asks the user how often he uses networked computers. He answers by selecting "rarely" from the presented menu of possible answers:

```
(interview-response :id 707 ... ((:answer "rarely") ... ))
```

Thereupon, the BGDL expression `(B S (network-novice *current-user*))` is passed on to the knowledge categorization component, which stores 'network-novice(*current-user*)' in partition SB. Now the stereotype activation condition for NETWORK-NOVICE is satisfied, so that SBUB is linked to the corresponding stereotype partition and inherits its contents, including the rule in item (3).

Moreover, the application informs BGP-MS about another mutually believed fact concerning domain objects, namely that the document the user is working on ('userdoc') is a plain text document:

```
(bgp-ms-tell (B S (B M (plain-text-document userdoc))))
```

BGP-MS enters 'plain-text-document(userdoc)' into partition SBMB. Since the contents of SBMB are inherited by SB, both this fact and the pre-defined fact 'Apple-LW(lw+)' (cf. item (5)) can be used for inferences in view SB as well. By hybrid reasoning, BGP-MS first infers that 'lw+' is a 'PS-Printer'. With the SB rule about printing (cf. item (4)), it can further conclude in SB that 'userdoc' cannot be printed on 'lw+' (see Section 4.5.1):

$\neg\text{printable-on}(\text{userdoc}, \text{lw+})$

In view SBUB, the rule inherited from NETWORK-NOVICE can be resolved with the pre-defined fact 'office-printer(*current-user*,lw+)' (which is inherited from SBMB, cf. item (5)) to

$\forall doc\ \ [\text{printed}(doc) \rightarrow\ \text{printed-on}(doc, \text{lw+})]$

During his session with the application system, the user is observed to invoke a print command on 'userdoc':

```
(d-act PRINT-COMMAND (userdoc))
```

The dialog act analysis component instantiates the presupposition schemes given in the definition of the PRINT-COMMAND dialog act (see item (6)) with the actual parameter 'userdoc', resulting in:

```
(bgp-ms-tell '(B S (B M (sent-for-printing userdoc))))

(bgp-ms-tell '(B S (B M (W U (printed userdoc)))))
```

Now the view-connecting rules (item (7)) are put into action (for more details concerning the next two steps see Section 4.5.2). The user goal just inferred from the PRINT-COMMAND dialog act along with the user rule that everything printed will be printed on 'lw+' (see above) match the premises of the view-connecting formula scheme. Hence the implicit goal that the user wants 'userdoc' to be printed on 'lw+' can be inferred:

$B_S W_U$ printed-on(userdoc, lw+)

Thus the content-specific view-connecting rule can "fire" and lead to the conclusion that the user believes that 'lw+' is able to print 'userdoc':

$B_S B_U$ printable-on(userdoc, lw+)

This is contradictory to what has just been inferred in partition SB. The resulting conflict between system's und user's beliefs can be detected and signaled to the application (cf. item (8)):

```
(alert-from-bgp-ms :misconception
  (B S (B U (printable-on userdoc lw+)))
  (B S (not (printable-on userdoc lw+))))
```

The application may now advise the user as to the incorrectness of his belief, and thereupon send the message

```
(bgp-ms-tell (B S (B U (not (printable-on userdoc lw+)))))
```

to BGP-MS if it assumes that the user has now corrected his misconception.

6 Interfaces for the Application Developer

BGP-MS provides several easy-to-use menu and graphics based interfaces that facilitate the definition of the structure and the content of user modeling systems for application developers. In this section we will briefly describe the graphical partition editor and the menu based stereotype editor. Both were developed using GINA [Spenke *et al.*, 1992], an object-oriented interface construction kit for OSF/Motif, written in Common Lisp and CLOS. A graphical editor for SB-ONE that is currently under development will not be discussed here since it forms an enhanced re-implementation of the SB-ONE editor described in [Kobsa, 1990; Kobsa, 1991]. All other parts of the user modeling system can be defined with a normal text editor using the textual definitions described in the previous sections. For some of these textual definitions, basic familiarity with Lisp is advantageous.

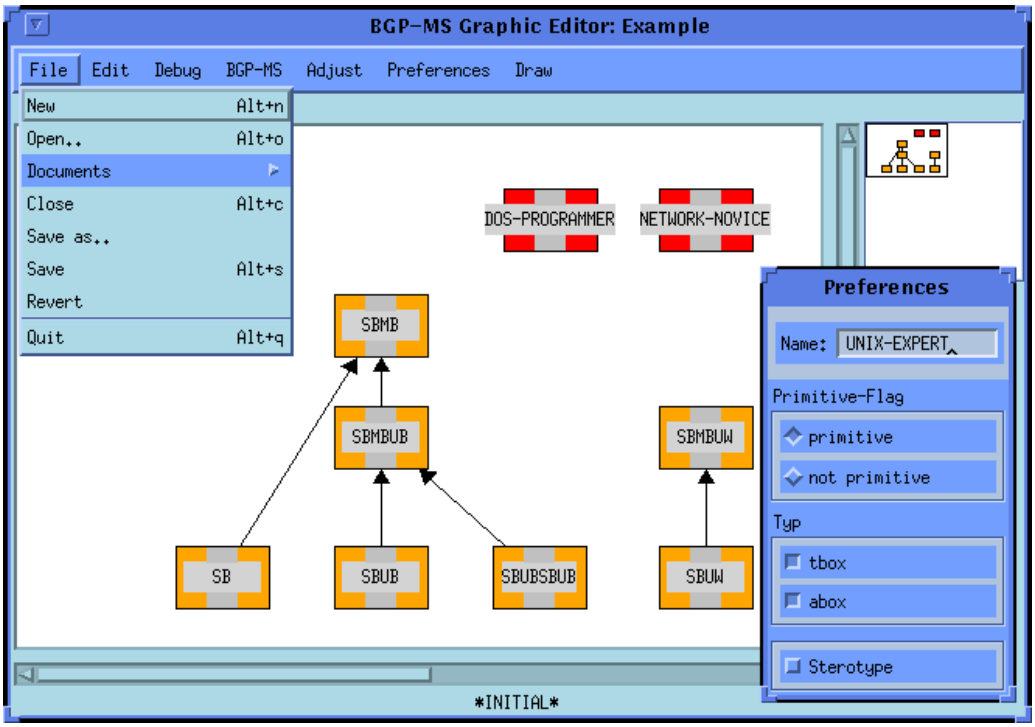

Figure 8: Graphical interface of the partition editor (original in color)

6.1 The partition editor

The graphical partition editor of BGP-MS [Knappe, 1994] allows the application developer to create and delete partitions, to introduce and delete inheritance relationships between partitions, and to assign certain properties to partitions (e.g., to name them or to declare them as stereotype partitions). Fig. 8 shows this interface with the example partition hierarchy that was already discussed in Fig. 3. Partitions become depicted as rectangular nodes, and inheritance relationships as directed links between nodes.

Stereotype partitions (in Fig. 8, DOS-PROGRAMMER and NETWORK-NOVICE) are marked out by a special color. All operations at the user interface are performed by direct manipulation in the central drawing area, or by the selection of menu items in pop-up and pull-down menus.

The central drawing area can be scrolled vertically and horizontally if the partition hierarchy becomes larger than the size of the window. Objects can be moved and repositioned for better layout. The right upper window in Fig. 8 shows the complete drawing area in miniaturized form. The rectangle in this window corresponds to the subarea that is currently visible in the main window. Moving this rectangle will change the visible subarea and allows one to quickly move to a different area of the partition hierarchy.

6.2 The stereotype editor

The stereotype editor of BGP-MS [Binder, 1994] has also been developed using GINA. It allows the application developer to define the activation and retraction conditions of the stereotypes of his application in a comfortable way. Fig. 9 shows an example with six stereotypes, whose names are listed in the upper left window. The stereotype NETWORK-NOVICE has been selected by the application developer for further inspection and definition. The four windows on the right side of the screen show the main contents of the selected stereotype, namely its concepts, its roles, its subsumption relations, and its FOPC formulas. The lower left quarter of the screen contains the windows of two "intelligent" text editors for the definition of stereotype activation and retraction rules, respectively. In the upper window one can see the activation condition for the selected stereotype NETWORK-NOVICE. This stereotype will be activated if the predicate 'network-novice' holds true about the user in partition SB (which contains the system's beliefs), or if none of the concepts 'LAN', 'ethernet' and 'file-server' are contained in partition SBUB. (The function IFUNKNOWN determines whether or not this is the case).

Activation and retraction rules must be Lisp expressions. In order to facilitate their definition, frequently-needed rule schemes have been pre-defined in BGP-MS. In Fig. 9, their names are listed in the second upper window from the left. The application developer can drag one or more of these names into a rule editor, whereupon a rule template will be pasted at the selected location (e.g. the template `(IFUNKNOWN '( ))`). Contents of the selected stereotype that are listed in one of the four windows on the right can likewise be dragged into the activation and retraction rule editors, and dropped at appropriate locations. Users may change existing rule schemes or define their own new rule schemes if needed, using the buttons on the right side of the rule scheme list and a pop-up editor.

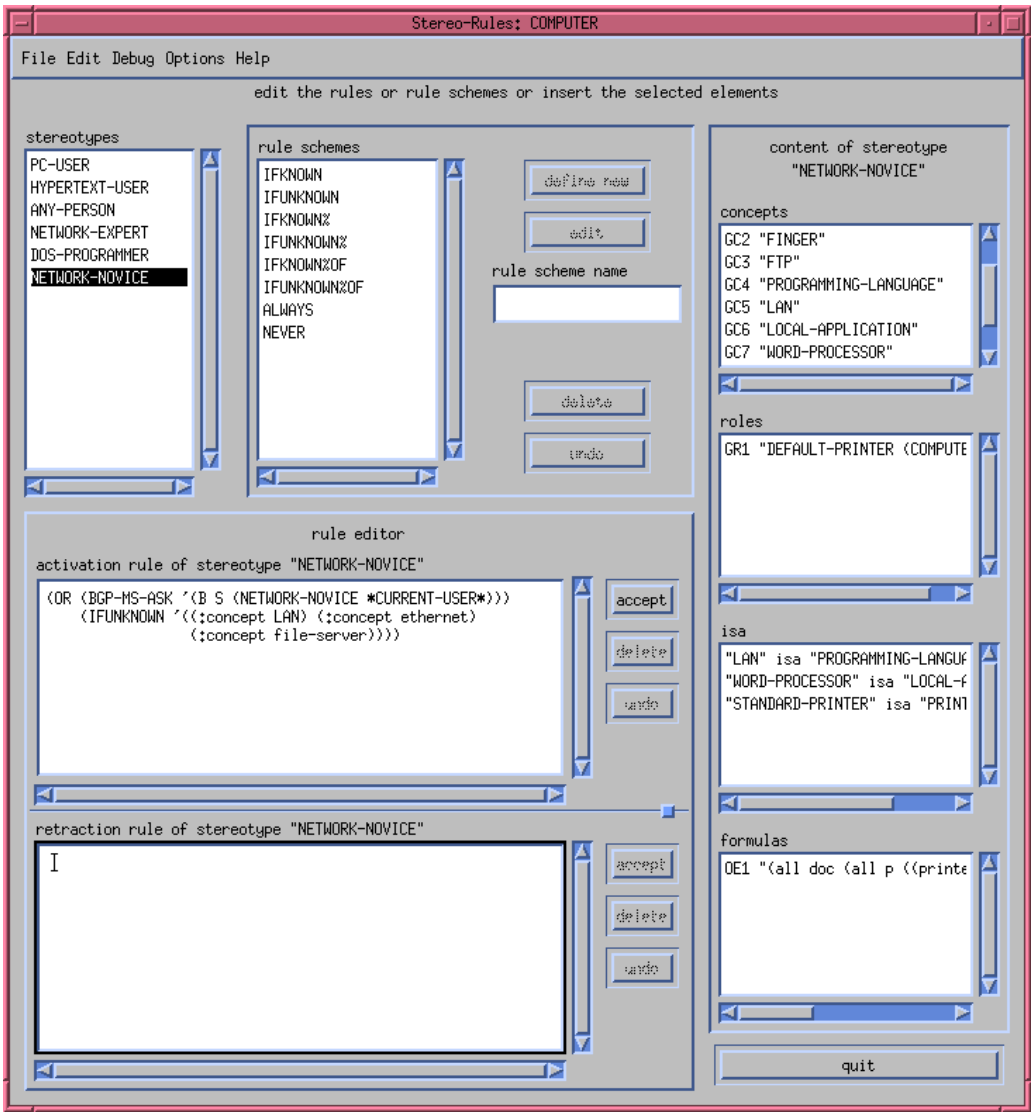

Figure 9: Graphical interface of the stereotype editor (original in color)

7 Comparison with other shell systems

In this section, the six major user modeling shell systems that have been developed to date will be surveyed and compared with BGP-MS. BGP-MS shares common aspects with all of these systems, but also contains many characteristics that are not present in most or all other shells. Such features include that BGP-MS allows the representation of more than one type of assumption about the user at the same time (e.g., both about his knowledge and about his goals); that it supports user interviews as well as inferences that are based on the user's actions as reported by the application; that it allows inferences about the user to be carried out concurrently with the operation of the application system; and that it is independent of the programming language in which the application has been written.

A wealth of features does however not imply better usefulness. As is standard practice in the selection of software tools, system developers should base the selection of a user modeling shell system on the demands of their applications. Advanced features of a shell may possibly require higher processing resources. For this reason, BGP-MS has been made tailorable, so that components can be deactivated or removed if they are not needed.

7.1 GUMS

The GUMS system [Finin, 1989], which is based on Prolog, is aimed at providing a set of services for the maintenance of assumptions about users' beliefs. GUMS does not draw assumptions itself. Instead, it accepts and stores new facts about the user which are provided by the application system, verifies the consistency of a new fact with the currently held assumptions by trying to deduce the negated fact from the current assumptions, informs the application system about recognized inconsistencies, and answers queries of the application concerning its current assumptions about the user. GUMS only allows one type of assumptions about the user to be made at a time (usually about his knowledge).

The shell system allows the definition of a stereotype hierarchy, whose topology is however restricted to a tree. Only a single stereotype may apply to the user at a time. The initial stereotype pertaining to the user must be selected either by the user model developer or the application system. Each stereotype includes a number of definite facts about the user; if one of these facts is contradicted by new information about the user, the stereotype is abandoned. It is then replaced by the most specific direct or indirect superordinate stereotype that does not contain the contradictory fact (stereotypes in parallel paths are not considered).

Two types of inference rules are supported by GUMS, namely definite rules and default rules. These are processed in a backward chaining mode when GUMS answers queries of the application system concerning the deducibility of a given fact from the current assumptions about the user, and when GUMS determines whether a new fact about the user is consistent with the current assumptions by trying to deduce the negation of this fact from its current assumptions. In both cases, negation as failure is possible if a predicate that has been asked for has been declared as closed, i.e., as defined by the current system knowledge pertaining to it. The use of default rules requires GUMS to continue searching for a goal until one is found that is not based on defaults, or until no more solutions exist.

GUMS offers an interesting mix of knowledge representation and inference methods for user modeling purposes, notably well-developed methods for default inferences. It is in many ways restricted, though. It only supports the modeling of one type of assumptions about the user, and the operation of the stereotype component is severely restricted. Since GUMS does not record the results of inferences, it must repeat the same inferences when the same query is posed again. On the other hand, however, it does not need a reason maintenance component like in BGP-MS. It remains to be determined whether the repetition of inferences or the truth maintenance management will cause higher costs in practical applications.

7.2 UMT

UMT [Brajnik and Tasso, 1992] allows the user model developer to define user stereotypes that contain the characteristics of user subgroups in the form of attribute-value pairs. Stereotypes can be ordered in arbitrary hierarchies which support the inheritance of stereotype contents. Each stereotype possesses an activation condition which specifies when the stereotype can be applied to the current user. UMT also puts a rule interpreter at the disposal of the user model developer which allows for the definition of user modeling inference rules. Possible contradictions between assumed user characteristics also have to be explicitly defined using rules.

Like GUMS, UMT does not draw assumptions itself, but accepts and stores new assertions about the user which are provided by the application system. Depending on the reliability of these assertions, they can be regarded as invariable premises or as (later still retractable) assumptions. Stereotypes that become activated due to new assertions about the user add still more assumptions, namely the attribute-value pairs describing the characteristics of the respective user subgroups. Some of the assumptions may possibly be contradictory. After stereotype activation, UMT applies all inference rules (including the contradiction detection rules) to the set of premises and assumptions, and records the inferential dependencies.

A reason maintenance component in the system then determines all possible user models, which are all consistent sets of assertions containing the premises, a subset of the assumptions, and all assumptions derived from these premises and assumptions. The current user model will be selected from the set of possible user models using preference criteria (e.g. assumptions which were reported to the system by the application have a higher weight than assumptions from stereotypes). If inconsistencies with new information from the application are later detected or if the application disconfirms advice from UMT concerning the user, the assumptions on which the offending assertion was based can be detected (since inferential dependencies became recorded), and the set of possible user models will be revised and re-evaluated to find the new current user model.

The strengths of UMT lie in its mechanism for recording the inferential dependencies between assertions and for determining the most preferable maximally consistent set of assumptions that can be maintained about the user at a given time. This mechanism is very time-consuming, however. Another possible obstacle to practical applicability are the limited representational and inferential abilities of UMT, which are based on attribute-value pairs and rules, respectively. A problem which UMT shares with the current version of BGP-MS is the fact that stereotype retraction is not integrated

into truth maintenance. The truth maintenance component can never decide to change the stereotypes that apply to the current user since stereotype activation and retraction is carried out by a completely independent component.

An interesting aspect in the comparison between UMT and BGP-MS is the fact that inference is exclusively carried out in a forward-chaining fashion in UMT, while it is bidirectional in BGP-MS. If forward chaining is employed, processing occurs when observations about the user are communicated by the application, but not when queries of the application have to be answered. If backward chaining is employed, processing is deferred to the time when queries have to be answered. If the application and the user modeling component are one single process, both strategies are problematic since the operation of the application may be retarded by the inferences in the user modeling component, either at entry time or at query time. If the user modeling component operates concurrently with the application (as is the case in BGP-MS, but not in UMT), then forward-chaining seems advantageous since it can be performed while the application is idle. Unfortunately, though, the number of inferred data also has to be taken into account: it may well be that forward chaining generates numerous inferences that will not be needed, but which nevertheless have to be managed by the truth maintenance component. BGP-MS therefore allows the application developer to restrict the depth of forward chaining, so that a compromise between space complexity and time complexity can be found that is appropriate for the specific application domain.

7.3 um

um [Kay, 1994]is a toolkit for user modeling whose architecture is simpler than those of other user modeling shell systems. However, it has been extensively tested empirically during the last few years, particularly in the context of a coaching system for a text editor, and a movie advisory system.

The basic knowledge representation elements in um are attribute-value-like structures named ‘components’. The system’s four different types of components represent items that the user may know, believe or prefer, or they represent arbitrary other user characteristics. Each component has the following information associated with it: a name, a verbal description, a type, a value, a time-stamp for the value, the source of the value, and additional information on the value. In the case of user preferences, a component can express whether or not the user prefers the component, is rather ambivalent, whether any information is available, or whether an unresolvable conflict occurred. In the case of beliefs and knowledge, a component can express whether or not the user knows/believes this item, does not want to know this item, or whether no information or only conflicting information is available.

A component is accompanied by a list of evidence, each of which may support the truth or falsity of the component. The source of the evidence, its type (observation, stereotype activation, rule invocation, entered by the user, told to the user), and its time stamp are also recorded. Explanations for components, sources of evidence, and types of evidence sources may be entered as well.

The user model structures can be accessed and evaluated by application-dependent and application-independent tools. The functionality of the tools that have been developed to date includes user

interviewing, knowledge elicitation, model verification by the user, user observation, and basic user model evaluation.

An interesting aspect of um's user models is that they are regarded as belonging to the respective users who can access them via a graphics-based hierarchical browser and, if they wish, modify them (this modification however does not actually override a user model entry, but is added as additional evidence). To facilitate this inspection, the components become organized in a hierarchy of topics and sub-topics (so-called 'partial models').

7.4 PROTUM

PROTUM [Vergara, 1994] is related to UMT except that it possesses more sophisticated stereotype retraction mechanisms than UMT. Most of the discussion in Section 7.2 about representational expressiveness, the separation of truth maintenance from stereotype activation and retraction, and the tradeoff between space complexity and time complexity also applies to PROTUM. Like UMT, PROTUM's truth maintenance mechanisms are far more developed than those of BGP-MS.

7.5 TAGUS

TAGUS [Paiva and Self, 1994] is a shell system for user modeling in interactive application systems, as well as for student modeling in intelligent tutoring systems and interactive learning environments. Like BGP-MS, it communicates with the application system by messages rather than function calls, but is not based on inter-process communication. Assumptions about the user are expressed using first-order formulas, with meta-level operators that denote different attitudes of the user towards these formulas (such as 'belief', 'goal', 'capability of reasoning').

TAGUS allows for the definition of a stereotype hierarchy and contains an inference mechanism, a truth maintenance system with different strengths of endorsements for assumptions about the user, and a diagnostic subsystem that includes a library of misconceptions. TAGUS also supports powerful update and evaluation requests by the application, including a simulation of the user (i.e., forward-directed inferences on the basis of the user model) and the diagnosis of unexpected user behavior.

7.6 DOPPELGÄNGER

A complementary yet partly competing proposal to user modeling shell systems are *user model servers*. User modeling shell systems, when filled with application-dependent user modeling knowledge, sort of become part of the application, i.e. receive information about the user from this application only and supply only this application with assumptions about the user. In contrast, user model servers are centralized user modeling components for all users and all applications on a network.

The DOPPELGÄNGER user modeling server [Orwant, 1991; Orwant, 1994] accepts TCP/IP connections from three different types of clients: hardware and software sensors provide data about users, applications employ DOPPELGÄNGER's assumptions for tailoring their behavior to users, and users can connect to the server directly if they wish to view or edit their user models. Various techniques for generalizing and extrapolating data from the sensors is put at the disposal of user model developers. These include beta distributions, linear prediction, and Markov models. Unsupervised clustering is available for collecting individual user models into so-called 'communities' whose information serves the purpose of stereotypes. (In contrast to all other user modeling shell systems, membership in a stereotype is probabilistic rather than definite). The different representations of DOPPELGÄNGER are quite heterogeneous.

User model servers seem advantageous for several reasons:

- The local processors that run the application program become relieved of user modeling tasks. This makes it possible to take advantage of user modeling even in devices with low computing resources (e.g., office machines).
- One may expect positive synergy effects from the combined results of several user-modeling applications. Data gathered from one application may possibly be useful for other applications as well. The consistency and coherence of users' work environments may thereby get improved.
- Updates in the user-modeling knowledge (e.g., new stereotypes or new inference rules) can be more easily performed on a central server rather than on numerous hosts for each user.

Central user modeling servers however also pose considerable security risks. DOPPELGÄNGER therefore takes advantage of an authentication server when processing messages from sensors, applications and users. Users modeled by DOPPELGÄNGER can control the access to any and all parts of their user model, tagging them entirely private or entirely public, or restricting the access to certain users. Users can also store their own user models on PCMCIA cards and remove them from the network.

To date, the networking and security features of BGP-MS are all provided by the host operating system. Transparent file access across the network allows one to use an adaptive application on any computer on the network, with only a single user model for each application. Access permissions allow the application developer to restrict read and write operations on the user model to processes owned by the user. Attempts will be made to enhance the communication architecture of KN-IPCMS to also allow for network communication. Additional security features will then be needed for guaranteeing the privacy of user models held by BGP-MS.

8 KN-AHS, an adaptive hypertext client of BGP-MS

8.1 Introduction

In this section we will describe the first application that takes advantage of BGP-MS, namely a hypertext system that adapts the presentation of its information to the presumed knowledge level of the user. The aim of this development was to demonstrate the feasibility of using BGP-MS in a normal hardware and software environment that is frequently found in the workplace. A more detailed discussion of this application can be found in [Kobsa *et al.*, 1994].

During the last few years, hypertext has become a very popular information structuring and presentation technique, e.g. for technical documentation and for information systems (see [Shneiderman and Kearsley, 1989]). A hypertext document consists of any number of objects that can be linked with one another in a network structure. The objects in a hypertext base are usually text documents, but can also include non-textual data like tables, graphs, animations, etc. (If an audiovisual component is involved, then the term 'hypermedia' is used). Hypertext is not necessarily read in a linear (i.e., sequential) order like conventional text, but can be read in a non-linear order by navigating within the hypertext node network. This non-linear linkage of objects represents the basic characteristic of hypertext.

Since hypertext is frequently read by users with differing knowledge and experience levels, it may at the same time be too difficult and too detailed for laypersons, and too redundant for experts. [Boyle and Encarnacion, 1994] showed empirically that an automatic adaptation of hypertext to the user's state of knowledge significantly improves text understanding as well as partially improving search speed. The system KN-AHS[8] adapts hypertext objects to the current user's state of knowledge. In contrast to other adaptive hypertext systems [Böcker *et al.*, 1990; Kaplan *et al.*, 1993; Boyle and Encarnacion, 1994; Beaumont, 1994], KN-AHS took advantage of an existing hypertext development product for the implementation of the hypertext and its user interface (namely TOOLBOOK [Asy, 1989]). In addition, KN-AHS employs the user modeling shell system BGP-MS for most of its user-modeling tasks.

8.2 Services of BGP-MS that are used by KN-AHS

Messages for reporting and querying assumptions about the user As will be explained in more detail below, KN-AHS monitors the user's interaction with the hypertext and occasionally makes the assumption that certain concepts in the domain of the hypertext are known or are not known to the user. It reports these assumptions to BGP-MS using `bgp-ms-tell` messages. Moreover, for appropriately adapting hypertext objects to the user's state of knowledge, KN-AHS needs to know whether or not certain concepts are probably known to the user. It obtains this information from BGP-MS using `bgp-ms-ask` messages.

[8] KN-AHS stands for KoNstanz Adaptive Hypertext System.

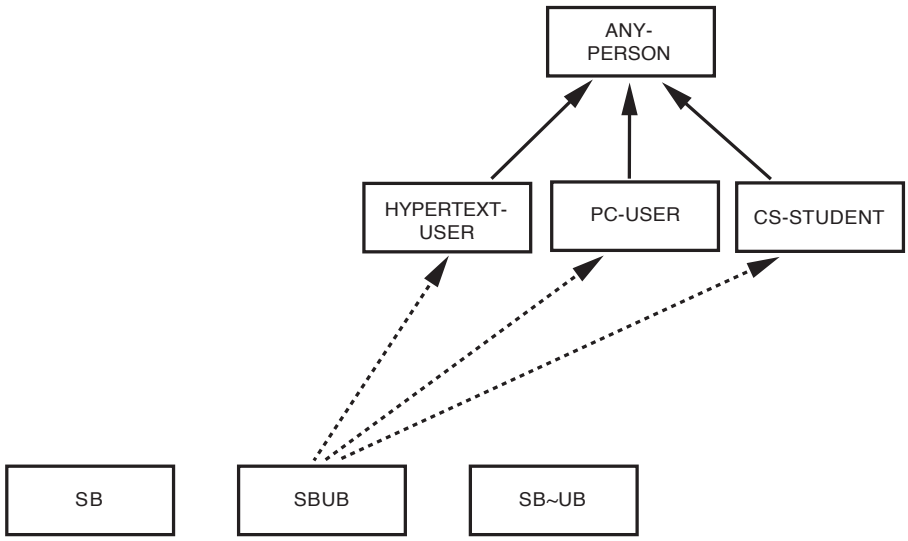

Figure 10: BGP-MS partitions used by KN-AHS

Partitions for the individual user model and the stereotypes Fig. 10 shows the simple partition hierarchy that is currently used for the purposes of KN-AHS. The depicted partitions can be divided into three groups:

- The *individual user model* consists of view SBUB, which contains BGP-MS's assumptions about the user's knowledge, and view SB∼UB, which contains BGP-MS's assumptions about what the user does not know[9].
- The *stereotypes* for user subgroups are ordered hierarchically. We assume for our application that a stereotype ANY-PERSON exists, which includes the knowledge that is available to any user. All other stereotypes include typical characteristics of users with various fields of specialization, namely hypertext users, PC users, and computer science students. They inherit the contents from the general stereotype.
- The *domain knowledge* of KN-AHS is included in partition SB.

The broken lines in Fig. 10 represent the possible inheritance relationships between partition SBUB and the stereotypes HYPERTEXT-USER, PC-USER and CS-STUDENT. Since only positive knowledge is contained in the stereotypes of KN-AHS at the time being, a link to SB∼UB is not possible. More than one stereotype can be active simultaneously.

SB-ONE for the representation of domain knowledge For the purposes of KN-AHS, partition SB becomes filled with terminological knowledge about the domain of the hypertext. This knowledge

[9] Assumptions about what the user does not know should actually be represented in modal logic (see Section 4.5.2). For reasons of simplicity and efficiency, the special partition SB∼UB has been introduced instead.

is exclusively represented in SB-ONE. Two types of SB-ONE concepts are distinguished: so-called 'field concepts', which represent small fields of knowledge, and 'terminological concepts', which represent technical terms. In Fig. 11, 'UM-Shell', 'GUMS', 'BGP-MS', 'KL-ONE', 'SB-ONE', 'CLASSIC' and 'BACK' are field concepts, 'role' and 'concept' are terminological concepts. The role 'associated-concept' (the only role used for KN-AHS) defines the relationship between fields and their associated terminology. The field concepts 'SB-ONE' and 'BGP-MS' in Fig. 11 inherit all roles of the field concept 'KL-ONE', and thereby all associated terminological concepts.

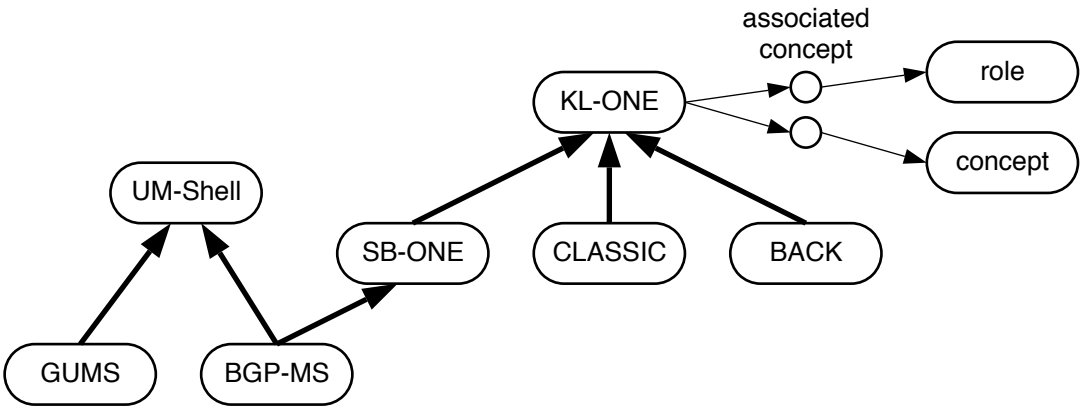

Figure 11: Detail of the concept hierarchy in partition SB used by KN-AHS

Domain-based inference rules KN-AHS exploits the fact that BGP-MS allows the definition of arbitrary inference rules (that become executed upon the receipt of `bgp-ms-tell` messages) for implementing rules that are based on the domain knowledge in partition SB. These rules take sub- and superfield relationships and the 'associated-concept' relationship between fields and their respective terminology into account, namely in the following way:

a) Sub- and superfield relationships

   1. If a minimum percentage P1 of direct subfields of a field were reported to be known/unknown, then all its direct subfields are known/unknown.
   2. If a minimum percentage P2 of direct subfields of a field were reported to be known/unknown (where P2 can be different from P1), then the direct superfield is also known/unknown.

b) Relationships between fields and their respective terminology

   1. If a minimum percentage P3 of the terminological concepts of a field were reported to be known/unknown, then all terminological concepts of the field are known/unknown.
   2. If a minimum percentage P4 of the terminological concepts of a field were reported to be known/unknown (where P4 can be different from P3), then the field is also known/unknown.

Conflicts can arise between the observations made by the application and assumptions inferred by these rules. If this is the case, then the inferred assumptions will have a lower priority and will be discarded from partition SBUB. Dependency management between premises and consequences cannot be considered at present since the truth maintenance component of BGP-MS is still under development.

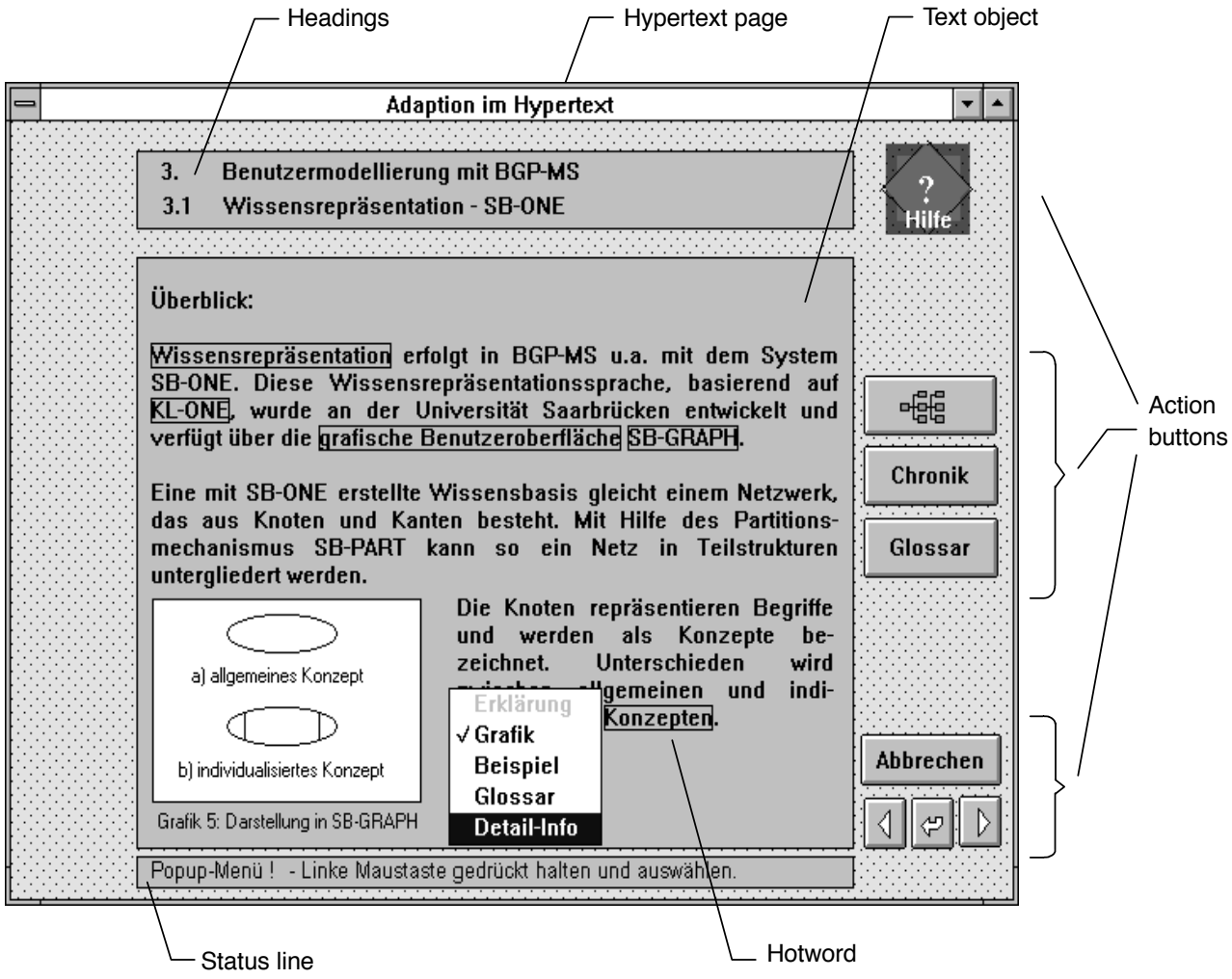

Figure 12: User interface of KN-AHS with selection menu for hotwords (original in color)

8.3 User model acquisition in KN-AHS

KN-AHS draws assumptions about the user's knowledge based on two information sources: an initial interview and some of the hypertext actions which the user may perform. In the interview (which is carried out without the help of BGP-MS), questions are posed to the user that refer to his membership in clearly separable user subgroups (like 'computer science student'), and his exposure to PCs, hypertexts, etc. The user's replies become communicated to BGP-MS, which can activate

initial stereotypes for him. If the user decides to skip this interview, BGP-MS will only activate the ANY-PERSON stereotype.

Certain actions that the user may perform afterwards in the hypertext give rise to assumptions about his familiarity with individual concepts. Fig. 12 shows the user interface of KN-AHS. The boxed words in the text are called *hotwords*. When the user moves the mouse cursor across a hotword, the form of the mouse cursor indicates the available actions for the hotword. The user may (a) jump to another text object that provides detailed information on the hotword; or (b) request additional information with a pop-up menu, namely a context-sensitive explanation (*Erklärung*), a graphic (*Graphik*), an example (*Beispiel*), a glossary definition (*Glossar*), or additional details (*Detail-Info*). In Fig. 12, the user just requested additional details for the hotword *Konzepten*.

KN-AHS draws the following primary assumptions about the user based on his actions at the interface:

- If the user requests an explanation, a graphic, an example or a glossary definition for a hotword, then he is assumed to be unfamiliar with this hotword.
- If the user unselects an explanation, a graphic, or an example for a hotword, then he is assumed to be familiar with this hotword.
- If the user requests additional details for a hotword, then he is assumed to be familiar with this hotword.

With each hotword about which more information can be requested, an SB-ONE concept that represents this technical term in BGP-MS is associated. When KN-AHS draws an assumption about the user's familiarity with a hotword, it notifies BGP-MS by a `bgp-ms-tell` message that the corresponding concept is known or unknown to the user. In the example in Fig. 12, the user will be assumed to be familiar with the hotword *Konzepten* since he requested additional details thereon. BGP-MS will therefore be notified that the user is familiar with the concept 'concept', which is associated with the hotword *Konzepten*. Concepts which the user knows or does not know will be entered into partitions SBUB and SB$\sim$UB, respectively. These primary assumptions will be compared with all stereotype activation and retraction conditions in regular pre-set intervals. As an effect, inheritance links between stereotype partitions and the partition SBUB may be added or deleted. In addition, the inference rules that are based on the domain knowledge in partition SB are tested and executed.

8.4 Adapting the document based on the user's conceptual knowledge

When the user switches to a new text object, KN-AHS aims at adapting it to his presumed conceptual knowledge. For each hotword in the new text object, KN-AHS asks BGP-MS about the user's familiarity with the corresponding SB-ONE concept using `bgp-ms-ask` messages. The hotword is then treated in the following way:

- If the user is unfamiliar with the associated concept, an explanation gets automatically added to the hotword. (The very same adaptation would take place if the user had manually selected

the ‘explanation’ entry in the pop-up menu for this hotword). Also, an icon that symbolizes an available graphic for the hotword is placed near the hotword.

- If the user is familiar with the hotword, more details are automatically added after the hotword.
- If no information is available from BGP-MS concerning the user’s familiarity with the hotword, then the hotword is not changed.

In the interaction between KN-AHS and BGP-MS, observations made by KN-AHS will be transferred asynchronously. This means that KN-AHS and BGP-MS run concurrently, so that the user model management will largely be performed while the user is reading the current text object. Questions posed to BGP-MS will however be handled synchronously and will have priority over incoming observations.

9 Summary and Implementation Details

In this paper we presented the user modeling shell system BGP-MS that supports developers in building user modeling systems for their applications. BGP-MS can help these applications cater to the current user's presumed knowledge, beliefs and goals. The system operates concurrently with applications. It offers applications several methods for communicating observations concerning the user, and for obtaining information on currently held assumptions about the user. It provides a choice of two integrated formalisms for representing beliefs and goals, and includes several types of inferences for drawing additional assumptions based on an initial interview, observed user actions, and stereotypical knowledge about pre-defined user subgroups. For tailoring BGP-MS to a specific application domain, the developer must select those components of BGP-MS that are needed in this domain and fill them with relevant domain-dependent user modeling knowledge.

BGP-MS has been implemented in CMU Common Lisp on SUN workstations [MacLachlan, 1992]. Runtime parts of BGP-MS have been ported to Golden Common Lisp 4.3 on the PC [Gol, 1992]. On SUN platforms, the graphical user interfaces have been developed using the construction kit GINA [Spenke *et al.*, 1992]. On the PC, some of these interfaces are currently being implemented in Microsoft's Visual C. KN-AHS has been developed on a PC platform under MS-DOS 6.2 and MS WINDOWS 3.1 using TOOLBOOK 1.5 [Asy, 1989]. A minimum memory of 32 MB (SUNs) and 16 MB (PCs) is recommended for running an application together with BGP-MS.

KN-IPCMS, the platform-independent communication protocol, has so far been implemented under UNIX System V and MS-DOS/WINDOWS 3.1. The UNIX implementation uses shared memory for data exchange and semaphores for coordination purposes. The current WINDOWS version of KN-IPCMS exploits the DDE functionality[10], which is also supported by TOOLBOOK and Golden Common Lisp.

BGP-MS is available by anonymous FTP from ftp.uni-konstanz.de.

Acknowledgment

The research described in this paper has been supported by the German Society for the Advancement of Scientific Research under Grant No. Ko 1044/4-2, and by the University of Konstanz under Grant No. AFF 17/92. We would like to thank Josef Fink, Jörg Höhle, Judy Kay, Detlef Küpper, Gloria Mark and the anonymous UMUAI reviewers for valuable comments on earlier versions of this paper.

[10] DDE (Dynamic Data Exchange) is a communication protocol that defines how WINDOWS applications can exchange messages and data, and helps application programs communicate with one another, as long as they can support DDE as well.

References

[Asy, 1989] Asymetrix Corporation, Washington, D.C. *Using TOOLBOOK: A Guide to Building and Working with Books (Version 1.5)*, 1989.

[Ballim and Wilks, 1991] A. Ballim and Y. Wilks. Beliefs, Stereotypes and Dynamic Agent Modeling. *User Modeling and User-Adapted Interaction*, 1(1):33–65, 1991.

[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.

[Binder, 1994] R. Binder. STED (StereoType EDitor): Ein Editor zur Erstellung und Verwaltung der Aktivierungs- und Deaktivierungsregeln von Stereotypen in BGP-MS. WIS Memo 13, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.

[Böcker *et al.*, 1990] H. Böcker, H.-D. Hohl, and T. Schwab. $\Upsilon\pi$ADAPT$\epsilon\rho$: Individualizing Hypertext. In *Human-Computer Interaction - INTERACT'90*, pages 931–936, Amsterdam, The Netherlands, 1990.

[Boyle and Encarnacion, 1994] C. Boyle and A. O. Encarnacion. MetaDoc: An Adaptive Hypertext Reading System. *User Modeling and User-Adapted Interaction*, 4(1):1–20, 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.

[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.

[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, 1978] P. R. Cohen. On Knowing What to Say: Planning Speech Acts. Technical Report 118, Department of Computer Science, University of Toronto, Canada, 1978.

[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, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1993.

[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.

[Gol, 1992] Gold Hill, Inc., Cambridge, MA. *GCLISP Developer 4.3 User's Guide*, 1992.

[Kaplan *et al.*, 1993] C. Kaplan, J. Fenwick, and J. Chen. Adaptive Hypertext Navigation Based on User Goals and Context. *User Modeling and User-Adapted Interaction*, 3(3):193–220, 1993.

[Kay, 1994] J. Kay. The um Toolkit for Reusable, Long Term User Models. To appear in *User Modeling and User-Adapted Interaction*, 1994.

[Kleiber, 1994] U. Kleiber. Erklärung in interaktiven Systemen und Unterstützungsmöglichkeiten durch das System BGP-MS. WIS Memo 6, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.

[Knappe, 1994] G. Knappe. Ein grafischer Partitionseditor für KN-PART, implementiert mit Hilfe der generischen Rahmenanwendung GINA. WIS Memo 14, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.

[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, 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 *Proc. of the Third International Workshop on User Modeling*, pages 35–41, Dagstuhl, Germany, 1992.

[Levesque, 1986] H. J. Levesque. Making Believers out of Computers. *Artificial Intelligence*, 30:81–108, 1986.

[MacLachlan, 1992] R. A. MacLachlan, editor. *CMU Common Lisp User's Manual*. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1992.

[McCoy, 1985] K. F. McCoy. Correcting Object-Related Misconceptions. Report MS-CIS-85-57, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1985.

[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.

[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.

[Orwant, 1991] J. Orwant. The Doppelgänger User Modelling System. In *Proc. of the IJCAI Workshop W4: Agent Modelling for Intelligent Interaction*, pages 164–168, Sydney, Australia, 1991.

[Orwant, 1994] J. Orwant. Heterogeneous Learning in the Doppelgänger User Modeling System. To appear in *User Modeling and User-Adapted Interaction*, 1994.

[Paiva and Self, 1994] A. Paiva and J. Self. TAGUS: A User and Learner Modeling System. In *Proc. of the Fourth International Conference on User Modeling*, pages 43–49, Hyannis, MA, 1994.

[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.

[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 *et al.*, 1995] W. Pohl, A. Kobsa, and O. Kutter. User Model Acquisition Heuristics Based on Dialogue Acts. In *Proc. of the International Workshop on the Design of Cooperative Systems*, pages 471–486, Antibes-Juan-les-Pins, France, 1995.

[Rich, 1979] E. Rich. User Modeling via Stereotypes. *Cognitive Science*, 3:329–354, 1979.

[Scherer, 1990] J. Scherer. SB-PART: Ein Partitionsverwaltungssystem für die Wissensrepräsentationssprache SB-ONE. Memo 48, Project XTRA, Department of Computer Science, University of Saarbrücken, Germany, 1990.

[Schiffer, 1972] S. R. Schiffer. *Meaning*. Clarendon Press, Oxford, 1972.

[Searle, 1969] J. R. Searle. *Speech Acts*. Cambridge University Press, 1969.

[Shneiderman and Kearsley, 1989] B. Shneiderman and G. Kearsley. *Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information*. Addison-Wesley, Reading, MA, 1989.

[Sitter and Stein, 1992] S. Sitter and A. Stein. Modeling the Illocutionary Aspects of Information-Seeking Dialogues. *Information Processing & Management*, 28(2):165–180, 1992.

[Sleeman, 1985] D. Sleeman. UMFE: A User Modelling Front-End Subsystem. *International Journal of Man-Machine Studies*, 23:71–88, 1985.

[Spenke *et al.*, 1992] M. Spenke, C. Beilken, T. Berlange, A. Bäcker, and A. Genau. GINA User Manual Version 2.1 for Common Lisp. Report 614, GMD, St. Augustin, Germany, 1992.

[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.

[Taylor and Carletta, 1994] J. A. Taylor and J. C. Carletta. Limiting Nested Beliefs in Cooperative Dialogue. In *Proc. of the 16th Annual Conference of the Cognitive Science Society*, pages 858–863, Atlanta, GA, 1994.

[Uehara, 1989] K. Uehara. An Intelligent On-Line Help System: ASSIST. *Future Generation Computer Systems*, 5(1):11–20, 1989.

[Vergara, 1994] H. Vergara. PROTUM: A Prolog Based Tool for User Modeling. WIS Memo 10, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.

[Winkels, 1990] R. Winkels. User Modelling in Help Systems. In D. H. Norrie and H.-W. Six, editors, *Computer Assisted Learning: 3rd International Conference*, pages 184–193. Springer, New York, 1990.

[Winkler, 1994] U. Winkler. Formalismentransformationen in KN-PART. Master's thesis, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.

[Winograd, 1988] T. Winograd. A Language/Action Perspective on the Design of Cooperative Work. *Human Computer Interaction*, 3(1):3–30, 1988.

[Wolz, 1992] U. Wolz. *Extending User Expertise in Interactive Environments: A Task-Centered Approach to Automatic Assistance*. PhD thesis, Department of Computer Science, Columbia University, New York, 1992.

[Zimmermann, 1994] J. Zimmermann. Hybride Wissensrepräsentation in BGP-MS: Integration der Wissensverarbeitung von SB-ONE und OTTER. WIS Memo 12, WG Knowledge-Based Information Systems, Department of Information Science, University of Konstanz, Germany, 1994.