Modeling Default Reasoning Using Defaults

PAUL VAN ARRAGON
Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
prvanarragon@watdragon.uwaterloo.ca

(Received 14 October, 1990; in final form 9 January, 1991)

Abstract. User modeling research can benefit from formal automated reasoning tools. However, existing formal tools may need to be modified to suit the needs of user modeling. Theorist is a simple framework for default reasoning. It can be used as a tool for building and maintaining a user model, and as a model of a user's default reasoning. To apply Theorist to both tasks, we develop Nested Theorist (NT), a simple tool based on Theorist that allows default reasoning on arbitrarily-many levels. We extend NT in two ways: we allow prioritized defaults, and we allow reasoning about agents with limited reasoning capabilities. This paper focusses on applications, and uses wide-ranging examples from user-modeling literature to illustrate the usefulness of the tools presented.

Key words: User Modeling, Theorist, default reasoning, nested reasoning, limited reasoning

1. Formal User Modeling Tools

Many issues that arise for user modeling also arise in the general context of knowledge representation, where they are dealt with abstractly by defining reasoning formalisms. Some of these formalisms can be applied to user modeling. For example, a key issue that arises for user modeling is how to build and maintain a detailed model of a user based on a small set of observations about the user. Abstractly, the problem is how to predict what is true about an object (the user) despite incomplete information about that object. One way to solve this abstract problem is to make assumptions about the object. This is the primary motivation for default reasoning research (Etherington, 1988).

Another key issue that arises for user modeling is how to reason about the beliefs and inferences of a user. To engage in dialogue with a user, it helps to know what the user believes, and what the user is able to derive from those beliefs. As user modeling research shows, it is useful to view a user as being able to reason with incomplete information (Joshi et al., 1984; Wilks and Ballim, 1987). The abstract problem of reasoning about belief has been dealt with by proposing formalisms (Konolige, 1982; Konolige, 1985; Levesque, 1984; Fagin and Halpern, 1988). However, few formalisms deal specifically with reasoning about a user (or any agent) that is reasoning with incomplete information.

User Modeling and User-Adapted Interaction **1:** 259–288, 1991.

This paper shows how to build a formal tool to deal with both of the above issues: reasoning with incomplete information, and reasoning about a user that is reasoning with incomplete information. The tool, called Nested Theorist (NT), provides a formal way to build and maintain a user model (by default reasoning), including a way to reason about what users believe and about how users reason by default.

An advantage of using a formal tool, such as NT, is that the input (domain knowledge) and output (user models) are given a clear semantics. This makes NT easy to use because statements of domain knowledge can be understood independently of each other, and the effect of individual statements on the whole system can be studied logically. A particular advantage of NT is that NT builds on research in the areas of default reasoning and reasoning about belief. Both of these areas have been studied extensively. NT enables this research progress to be applied to user modeling.

Another advantage of NT is that NT makes a clear distinction between levels of reasoning. In NT, both the system (that is reasoning about the user) and the user (who is reasoning about some domain) have incomplete knowledge, and use defaults. Hence defaults occur on multiple levels in NT. Although some user modeling research studies agents that use defaults (Joshi et al., 1984; Perrault, 1988; Wilks and Ballim, 1987; Wilks and Bien, 1983) the level distinction is not fully made. An advantage of making a clear distinction is that a wide range of user modeling research can use the NT framework. We illustrate this by implementing examples drawn from various existing user modeling systems.

1.1. Context

Figure 1 illustrates this paper's view of the context of a user model. At the core of the computer system is the belief subsystem, which maintains and reasons with the system's beliefs. Initially, a knowledge base (KB) is input by the KB designer. A fully-integrated subpart of this KB is the user model (UM). The UM is designed to contain general statements about users, and to accumulate specific statements about specific users during dialogue with users. According to a formal definition of reasoning (symbolized by $\models$), the belief subsystem is able to predict other beliefs (KB+) from the given KB, including more beliefs about users (UM+). For example, in NT, $\models$ is a type of default reasoning.

The computer user is able to communicate with the system by asking questions (Q) and providing answers (A). This is done through an interface that uses a theory of dialogue to generate helpful answers and ask appropriate questions. Sometimes the answers provided by the user to the interface are

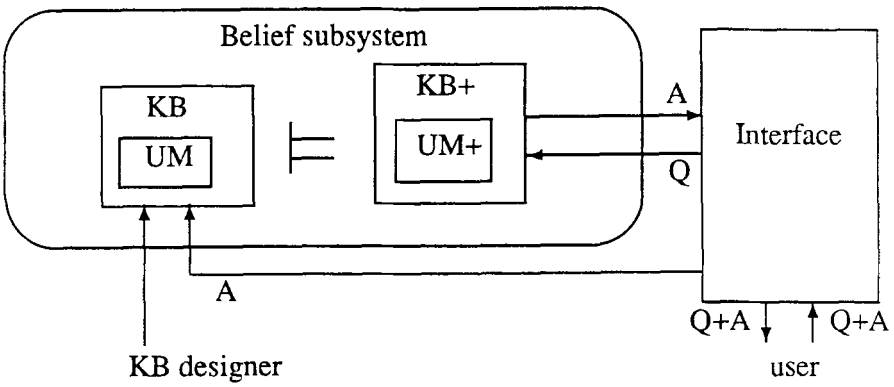

Fig. 1. The context of a user model.

added to the KB. In this way, the system accumulates specific knowledge about specific users.

When the interface requires information about the domain or about the user, it poses questions that are answered by the belief subsystem. The belief subsystem is driven by these questions. Assuming the interface only asks specific questions, such as whether a formula α is in KB+, the belief subsystem needs only to be able to compute specific answers, rather than be able to derive the complete KB+.

Given this context, a user modeling system consists of four components:

1. an underlying formal definition of reasoning ($\models$),
2. an implementation of the definition,
3. a method for KB designers to organize the KB within the definition, and
4. a fully compiled KB organized according to the method.

Components 1 and 2 together define a user-modeling tool. Such a tool (often called a shell) consists of a definition of reasoning (including reasoning about the user model) and a domain-independent implementation of this definition of reasoning.

Every user modeling system must incorporate some underlying form of reasoning about a user. The reasoning may not be completely formalized, but some reasoning must be used. The availability of a formal user-modeling tool frees a researcher to work on issues regarding how to structure knowledge (component 3), and what knowledge to use (component 4), without having to formally define the underlying reasoning, or having to consider implementation details.

The definition of reasoning (component 1) provides ways to deal with particular abstract issues that arise. For example, the tools of this paper provide a way to reason with incomplete knowledge, and a way to reason about users who have incomplete knowledge and who have limited reasoning capabilities. The capabilities of the tools suggest ways of structuring knowledge (component 3) without prescribing exactly how knowledge is to be structured. For example, the tools presented in this paper are capable of default reasoning, which enables incorporating many user-modeling techniques: using stereotypical knowledge, attributing capabilities to users, ascribing belief to users, using user utterances to modify a model, and deciding what to say to a user. None of these techniques are inherent in default reasoning, however. The KB designer must decide how to use the tools for a given application.

1.2. Design Decisions

The following summarizes our design decisions for our user modeling tools. The first eight concern the formal definition of nested reasoning; the last one concerns the implementation.

Decision 1. *Reasoning is nested to arbitrary depth.*

For generality, the tool is not limited to two levels of nesting, but allows an arbitrary number of levels. This allows the system to reason about users who are reasoning about other agents.

Decision 2. *There may be many agents on each level (except on the top level).*

Also for generality, the tool may have *many* object-level reasoners (users), and many agents on each more-deeply-nested level. In user modeling, there may be more than one user. Each user may reason about several other agents, each of these agents may reason about several other agents, and so on. However, we need only have one agent on the top level, since the top level corresponds to the system, and in user-modeling applications there is only one system involved that reasons about the users.

Decision 3. *An agent's representation of another agent is incomplete.*

The system does not have complete information about the user, and the user may not have complete information about other agents. Our language should allow representing incomplete knowledge.

Decision 4. *Each level supports deduction.*

The system can draw inferences about the user, and model that the user draws inferences. We use logic. It may be argued that logic is a poor representation of the nuances of human reasoning. Our view is that logic is useful for user modeling in the same way that logic is useful for other domains (Hayes, 1977). Logic provides a way to talk about knowledge precisely

with a straightforward declarative semantics, and a powerful representation language.

We do not commit to logic in the strong sense of claiming that a user reasons according to deduction in some logic. Logic provides a way to represent inferences of a user, but does not completely specify how the inferences are drawn.

Decision 5. *Each level supports default reasoning.*

To overcome the rigidity of deduction, the system models the user as able to make assumptions. The system can also make assumptions about the user, such as that the user can draw an inference.

Defaults have been shown to be useful for performing a wide variety of types of reasoning. Whereas deduction is sound inference in a logic, reasoning by default is unsound inference. This unsound default inference can take many forms, such as analogy (Goebel, 1989). Defaults can be used to reason despite incomplete information. Furthermore, defaults may have associated priority levels that allow a more flexible type of reasoning (Brewka, 1989). Hence our commitment to logic does not pose as strong a limitation as at first it seems.

Decision 6. *Knowledge and belief are not defined.*

Instead of defining the notion "belief" or "knowledge," the tool is based on the notion "what is derived by an agent." In this view, a user has given premises with which to derive beliefs. Representing this reasoning procedure enables using the system to predict what a user will infer from what the system tells the user, and how a user will revise her beliefs.

Because each user may be different, there is no fixed logic of belief. Instead, there is only a representation of a reasoning procedure. This reasoning procedure is not completely fixed either, because the system can reason about limitations of the user's ability to apply the reasoning procedure.

Knowledge is often defined as true justified belief (Shoham and Moses, 1989). The tool need not incorporate any such definition. Instead, the tool models the argument used to derive a "belief." These arguments have various properties: some are based on premises that the user believes firmly (facts), and others are based on premises that are assumed more tenuously (defaults). The former arguments correspond more closely to knowledge, and the latter correspond more closely to beliefs.

Decision 7. *The representation is general.*

The tool functions as a programming language for applications that require a model of belief. A general representation allows many techniques to be incorporated, and does not limit the domain of the KB. Logic is advantageous because it is a well-understood and general representation, thus logic simplifies the amalgamation of specific application ideas under a unified framework.

Decision 8. *The tool does not incorporate psychological theories.*

Representing what a user can derive seems to rely heavily on psychology. Cognitive psychology researches what people typically believe, how people change their minds when faced with conflicting evidence, and how learning takes place (Hoenkamp, 1987). Although each of these issues is relevant to user modeling, this paper avoids these issues. No claim is made about the psychological validity of the tools presented.

Our view is that the formal aspects and the psychological aspects of user modeling are not in competition. One does not have to choose between a psychological representation and a logical representation. Instead, a formal logical tool presents a language that can be used to express relevant psychological theories.

A formal definition makes a tool easier to use. Rather than having to design ad hoc structures when implementing a technique based on psychology, a KB designer can structure psychological ideas according to a principled tool. For this to be possible, the tool must be flexible, not enforcing a particular psychological view, and allowing a flexible language for expressing knowledge. For example, if the tool is to be used for reasoning about a user's limited ability to reason, the limitations should not be fixed by the tool, but the tool should provide a way to express and reason about such limitations. We achieve this by using a logical language that can express defaults.

Decision 9. *The tool is implemented efficiently, but without insuring worst-case tractability.*

A possible problem with generality is that there is potential for intractability (Levesque and Brachman, 1985). Reasoning with a full first-order predicate calculus reasoner is known to be intractable in general. Adding defaults increases the potential for intractability (Reiter, 1980). However, a tool should be general, in the same way that a programming language is. Rather than restricting the use of a tool to efficiently-solvable problems, we leave the tool general so more difficult problems can be represented and reasoned about.

In this view, tractability is a property of the application, rather than of the tool. Specific instances of average problems that arise may typically be tractable enough (Halpern and Moses, 1985, p. 489). The tool is general, but finds answers to specific problems efficiently. Overhead arising from the implementation is small, so the time and space required to solve a problem depends mainly on the complexity of the specific problem.

2. Theorist

Theorist (Poole et al., 1987; Poole, 1988) is a simple framework for default reasoning. Its input is two sets of formulae, called facts and defaults. Theorist

uses facts and consistent defaults as premises in a logical argument. If a formula g is a logical consequence of the facts and consistent defaults, we say that Theorist can *explain* g from the facts and defaults. If new facts are added later, it may be that Theorist can no longer explain g because the defaults used are inconsistent with the new facts.

Explanations in Theorist are defined in terms of two sets of formulae input by the knowledge designer: [1]

> $\mathcal{F}$ a set of facts: closed formulae taken as true in the domain;
> Δ a set of defaults: (possibly open) formulae taken as the "possible hypotheses" in the domain.

DEFINITION 2.1. *An* **explanation** *from $\mathcal{F}$, Δ of a closed formula g is a set $\mathcal{F} \cup D$ where D is a set of ground instances of elements of Δ such that $\mathcal{F} \cup D$ is consistent and implies g.*

That is, from $\mathcal{F}$, Δ, Theorist can **explain** g if there exists a set D such that

1. $\mathcal{F} \cup D \models g$,
2. $\mathcal{F} \cup D$ is consistent, and
3. D is a subset of ground instances of elements of Δ .

3. Theorist as a User Modeler

To illustrate Theorist, and to show how Theorist can be applied to user modeling, we consider some simplified examples from user modeling literature. In our notation, variables are upper case letters. Functions, predicates, and constants are strings beginning with a lower case letter. Variables are sometimes universally quantified using $\forall$. The following logical connectives are used: implication ($\leftarrow$), conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$).

EXAMPLE 3.2. *UNIX Novices*

In a system that provides consultation for users of the UNIX [2] operating system, it is useful to model the level of expertise of the user (Chin, 1988, 1989). This expertise can take the form of a default. For example, we may assume that novice [3] users understand the simple command "login." If we

[1] Note that throughout this paper, we use the word "explains" because it corresponds best to Theorist terminology. However, the propositional attitude we intend to express more closely resembles "predicts" or "believes".

[2] UNIX is a trademark of Bell Labs

[3] We ignore for now the issue of how these stereotypes are obtained.

have a fact that David is a novice, Theorist can explain that David understands the "login" command.

$$\mathcal{F} = \{\ novice(david)\ \} \tag{1}$$
$$\Delta = \{\ understands_A(login) \leftarrow novice(A)\ \} \tag{2}$$

From the above fact and default, $\mathcal{F} \cup \{understands_{david}(login) \leftarrow novice(david)\ \}$ is an explanation of $understands_{david}(login)$.

An explanation can be blocked by adding facts that contradict defaults used in the explanation.

EXAMPLE 3.3. *Blocking an Explanation*

If the system observes that Eric is having trouble logging in, it may add this observation as a fact. Given that the system has a fact that users who cannot login do not understand the "login" command, the system cannot explain that Eric understands the "login" command.

$$\mathcal{F} = \{\ novice(eric) \tag{3}$$
$$cannotLogin(eric) \tag{4}$$
$$\forall A\ \neg understands_A(login) \leftarrow cannotLogin(A)\ \} \tag{5}$$
$$\Delta = \{\ understands_A(login) \leftarrow novice(A)\ \} \tag{6}$$

From the above, $understands_{eric}(login)$ is not an explanation since $\{\ understands_{eric}(login) \leftarrow novice(eric)\}$ is inconsistent with $\mathcal{F}$.

Rather than have a default such as $understands_A(login) \leftarrow novice(A)$ for each UNIX command, we can organize these defaults usefully by classifying each command according to whether it is mundane or simple. (Chin, 1988, 1989, calls this a double stereotype.) This technique allows us to specify the defaults more compactly.

EXAMPLE 3.4. *Double Stereotypes*

UNIX commands are classified by the facts as being simple or mundane. The defaults specify that novices understand simple commands, and that beginners understand simple and mundane ones. If we have as a fact that a user is a novice or a beginner, we can assume the user understands all commands of the appropriate category.

$$\mathcal{F} = \{novice(david) \tag{7}$$
$$beginner(kathy) \tag{8}$$

$$simple(login)) \tag{9}$$
$$simple(rm)) \tag{10}$$
$$mundane(ed)) \tag{11}$$
$$mundane(mv)) \tag{12}$$
$$mundane(cp)) \quad \} \tag{13}$$
$$\Delta = \{understands_A(C) \leftarrow novice(A) \wedge \tag{14}$$
$$simple(C)$$
$$understands_A(C) \leftarrow beginner(A) \wedge \tag{15}$$
$$(simple(C) \vee mundane(C)) \ \}$$

From the above, $understands_{david}(C)$ is explainable if C is one of $\{login, rm\}$, and $understands_{kathy}(C)$ is explainable if C is one of $\{login, rm, ed, mv, cp\}$.

The expressability of the underlying first-order logic that Theorist uses is an advantage. For example, we can express disjunctive knowledge. We may have a fact that u is a novice or a beginner, but not know which.

EXAMPLE 3.5. *Disjunctive Stereotype*

Let us add to the facts of example 3.4 a fact that Eric is a novice or a beginner.

$$(beginner(eric) \vee novice(eric)) \in \mathcal{F} \tag{16}$$

From the above, we can explain that $eric$ understands the login command, since both beginners and novices understand simple commands. Note that we do not have to first eliminate all but one of the stereotypes, as in (Chin, 1989), before we can derive $understands_{eric}(login)$. In a sense, extra expressability provides a computational advantage.

Theorist is a flexible framework. Following design decision 7, Theorist does not specify exactly how knowledge must be structured. For example, the idea of organizing knowledge into double stereotypes is orthogonal to Theorist. For another example, stereotypes can be structured into a hierarchy, as is (Finin and Drager, 1986), to enable reasoning about how a stereotype inherits properties of more general stereotypes. Nothing extra is needed in the definition of Theorist to make this reasoning possible, since Theorist defaults can be organized in a hierarchy (Jones and Poole, 1985).

Furthermore, stereotypes need not be specified as facts as in the above examples. Instead, we can use observations to derive the user type, as Chin does. The basic structure of domain knowledge would be as follows:

observations →stereotype →predictions

For example, instead of having "novice(david)" as a fact, we may have observations about commands that David used in the past. From these observations, we use facts or defaults to derive what type of user (stereotype) David is. For example, we may have a fact that users who use the "ed" command are either novices or beginners, and a default that users who use the "ed" command are beginners. Once we derive a stereotype from these observations, we can derive further predictions about the user.

The above examples illustrate that, despite not being originally intended to be applied to user modeling, Theorist is applicable to user modeling because it provides a way to reason with incomplete information. We need only specify as facts and defaults the relevant knowledge about observations, stereotypes, and predictions.

4. Theorist as a Model of Reasoning

Much of the impetus for user modeling comes from the desire to improve communication between system and user. The capability of modeling the level of competence of the user (as in the above examples), can help the system to decide what advice is appropriate to the user (Chin, 1988, 1989; Finin and Drager, 1986).

Sometimes a more detailed model of the user is helpful. To understand and respond to a user, it is helpful to have a model of what the user *believes*. For example, the user may not merely understand the "rm" command, but may also believe particular details about how the "rm" command was used during the current session. If the system knows what the user believes, the system can avoid telling the user details the user already believes, but can tell the user details that are important for the user's goals (Cohen et al., 1989).

Furthermore, it is helpful to model how the user *reasons*. If the system knows how the user reasons, the system can tell the user just the necessary detail so the user can derive the rest herself. A model of how the user reasons also enables the system to prevent the user from drawing false conclusions (Joshi et al., 1984).

Theorist does not have any built-in capabilities of modeling how a user reasons. So to apply Theorist to modeling a user's reasoning, we must augment Theorist with a model of reasoning. But what capabilities should the model of reasoning have?

It would be useful to model a user as at least being able to make deductive inferences. For example, if we tell the user that "rm t*" removes files that begin with the letter "t," we can reason that the user can deduce that "rm t*" will remove a file called "temporary."

It has been argued in (Joshi et al., 1984) that it is also useful to model assumptions the user might make. Since the user also has to deal with incomplete information, the user may assume things that are typically true. The system should block any of these assumptions that are false. For example, the user might assume that a command does not have negative side effects if there is no error message. If we tell the user that "rm t*" removes the file named "temporary," the user may assume that "rm t*" has no negative side effects. We should warn the user that "rm t*" removes *all* files with names that begin with the letter "t," such as the file named "thesis."

This leads to the idea of employing Theorist as a model of the user's reasoning. We can view a user as having a set of facts and defaults in her mind, and using these to derive beliefs. Technically, a belief is viewed as any formula that can be explained from a set of facts and defaults. This is not to take the view that people are actually Theorist systems, but that a Theorist system is a useful model of some parts of a person's reasoning.

5. Nested Theorist

The tool we present has Theorist on two levels. To obtain a nested formalism, we axiomatize one Theorist system in the language of another Theorist system. To do so, we need to define a language in which to axiomatize Theorist using facts and defaults.

5.1. Meta and Object Language

Suppose a system, s, is modeling a user, u. On the metalevel, Theorist is used to build and maintain a model. This corresponds to s reasoning about u. On the object level, Theorist is used as a model of how u reasons. This corresponds to s's model of u's reasoning about the world.

Technically, we can achieve this by defining two languages: an object language (OL) to express u's facts and defaults, and a metalanguage (ML) to express s's facts and defaults (Konolige, 1982). The OL is part of the object of study of the ML, since the ML is be able to refer to u's facts and defaults that are expressed by the OL.

Both the ML and OL are first-order languages. That is, they consist of constants, functions, predicates, variables, connectives, and quantifiers, combined to form formulae, and given a Tarskian semantics. [4] To refer to statements in the OL, the ML has terms that denote these sentences. For each OL constant and variable, the ML has a corresponding constant. For each OL function, predicate, connective, and quantifier, the ML has a corresponding function.

[4] See (van Arragon, 1990) for details.

This is summarized in table I, where the OL predicate r is represented by the

OL sentences	*ML terms*
$r(f(a))$	$r'(f'(a'))$
$p \leftarrow q$	$if(p', q')$
$p \wedge q$	$and(p', q')$
$p \vee q$	$or(p', q')$
$\neg p$	$not(p')$
$\forall X\ r(X)$	$forall(x', r'(x'))$

TABLE I
Representing OL in ML

ML function r'; the OL function f is represented by the ML function f'; the OL constant a is represented by the ML constant a'; the OL predicates p and q are represented by the ML 0-ary functions p' and q' respectively; and the OL variable X is represented by the ML constant x'. We do not discuss here many of the technical issues that arise in a ML/OL structure. (See Konolige, 1982; van Arragon, 1990). Rather, we focus on the aspects that are unique to NT.

To refer to u's facts, defaults, and explanations, the ML has functions $\mathcal{F}_u$, Δ_u, and E_u^D, where D is replaced with a set of defaults used in the explanations. The ML can express that a statement is in the facts of the user using the predicate "$\in$." For example, $p'(a') \in \mathcal{F}_u$ is a ML statement expressing that $p(a)$ is a fact of u.

Notation

For ease of notation, we do not use the $\in$ predicate, but instead treat $\mathcal{F}_u$ as a predicate. For example, to say that $p(a)$ is a fact of u, we state $\mathcal{F}_u(p'(a'))$. Similarly, we treat Δ_u and E_u^D as ML predicates in our notation. $\Delta_u(p')$ means that p is a default of u, and $E_u^{\{p'\}}(q')$ means that q is explained by u, using default p.

Furthermore, instead of using the ML with all of its complexity, we replace ML terms with the corresponding OL sentence. For example, rather than stating

$$\mathcal{F}_u(\ forall(x', and(p'(x'), q'))\ ) \tag{17}$$

to say that $\forall X\ p(X) \wedge q$ is in the facts of u, we use the following simpler notation:

$$\mathcal{F}_u(\ \forall X\ \ p(X) \wedge q\ ) \tag{18}$$

This is a notational convenience only. The ML is still actually first-order.

5.2. Assuming Consistency

Imagine a scenario where the system s is reasoning about a user u and the assumptions u can make. If s builds a model of u's reasoning where u makes assumptions to explain some formula g, there are two conditions this model should satisfy (see definition 2.1):

Condition 1: the assumptions (together with the facts) of u imply g, and

Condition 2: the assumptions of u must be consistent with the facts of u.

To show that condition 1 holds, s must show that a subset of u's facts ($\mathcal{F}_u$) together with the assumptions u makes imply g. It suffices to show that a subset of $\mathcal{F}_u$ implies g because such deductive conclusions are *monotonic*. That is, given that $\mathcal{F}_u^*$ is a subset of $\mathcal{F}_u$, and that D is the set of assumptions of u, if $\mathcal{F}_u^* \cup D \models g$ then $\mathcal{F}_u \cup D \models g$.

To show that condition 2 holds would require knowledge of the *complete* set $\mathcal{F}_u$, which s does not have. If s discovers a new element of $\mathcal{F}_u$, then u's assumptions may no longer be consistent since default conclusions are *nonmonotonic*. That is, given that $\mathcal{F}_u^*$ is a subset of $\mathcal{F}_u$, and that D is the set of assumptions of u, if $\mathcal{F}_u^* \cup D$ is consistent, it does not necessarily follow that $\mathcal{F}_u \cup D$ is consistent.

Consider the following example, in which s has two facts: that u has a fact b, and that u has a default $f \leftarrow b$. [5]

EXAMPLE 5.6. *Unsatisfied Condition 2*

The following are facts of s:

$$\mathcal{F}_u(b) \tag{19}$$

$$\Delta_u(f \leftarrow b) \tag{20}$$

s cannot explain that u explains f, because s cannot show that $f \leftarrow b$ is consistent with u's facts. (19) does not say that b is the *only* fact of u. There may be other unknown facts of u that are inconsistent with $f \leftarrow b$. For example, u may have the fact $\neg f$.

The inability to satisfy condition 2 stems from incomplete knowledge regarding the facts of u. By design decision 3, we allow situations where s is unable to obtain complete knowledge of u. Fortunately, we can deal

[5] This is a propositional version of the well-known birds fly example.

with this inability using the technique Theorist already uses to reason despite incomplete knowledge. That is, we can use a default. We specify that for any set of assumptions D, provided that s cannot show that $\mathcal{F}_u \cup D$ is inconsistent, s can assume that $\mathcal{F}_u \cup D$ is consistent. We call this the Consistent Assumptions Default (CAD). CAD leaves open the possibility that other facts exist that s does not know about, but still enables s to reason that u can consistently use defaults.

Reconsider example 5.6. Assuming $\mathcal{F}_u$ is consistent with default (20) enables s to satisfy condition 2. Hence, s can show that u explains f, as desired, and it is still possible that u has facts other than b, as long as they do not conflict with (20). For example, the assumption of consistency precludes that u has a fact $\neg f$, but u may have any number of facts unrelated to f and b.

DEFINITION 5.7. **NT** *is a Theorist system with a ML that refers to an OL (as described above) such that the metalevel Theorist, s, has a fact that the user, u, forms explanations according to the definition of Theorist:*[6]

$$\forall D \forall G \;\; E_u^D(G) \leftarrow \mathcal{F}_u \cup D \models G \;\wedge\; \mathcal{F}_u \cup D \text{ is consistent} \;\wedge\; D \subseteq \Delta_u \tag{21}$$

and a default that the user's assumptions are consistent. That is, s has the following default:

$$(\mathcal{F}_u \cup D \text{ is consistent}) \leftarrow (D \subseteq \Delta_u) \tag{CAD}$$

(van Arragon, 1990) shows how this definition can be altered to allow reasoning about more than one user, and to allow reasoning about how u reasons about other reasoning agents.

6. NT Examples

To illustrate the properties of definition 5.7, and how it can be applied to user modeling, we consider several examples. In these examples, we list all relevant facts and defaults of s. Note however, that we do not rewrite the fact and default of s that occur in definition 5.7. These are operative in the examples, but left implicit.

[6] Some details are left out here. See (van Arragon, 1990).

EXAMPLE 6.8. *Reasoning about Defaults*

s has two facts: that u has a fact that no error message occurs after the command "rm t*," and that u has a default that if a command has bad side effects there would be an error message.

$$\mathcal{F}_u(\ \neg errorMessage(\text{“}rm\ t^*\text{”})\ ) \tag{22}$$

$$\forall C\ \Delta_u(\ errorMessage(C) \leftarrow sideEffects(C)\ ) \tag{23}$$

From (22) and (23), we can show that s can explain $E_u^D(\neg sideEffects(\text{“}rm\ t^*\text{”}))$ with $D = \{errorMessage(\text{“}rm\ t^*\text{”}) \leftarrow sideEffects(\text{“}rm\ t^*\text{”})\}$. To see this, consider the three conjuncts in statement (21) of definition 5.7 with D as above, and $G = \neg sideEffects(\text{“}rm\ t^*\text{”})$. The three conjuncts are

Conjunct 1: $\mathcal{F}_u \cup D \models G$

Conjunct 2: $\mathcal{F}_u \cup D$ is consistent

Conjunct 3: $D \subseteq \Delta_u$

They can be explained by s as follows:

1. By (22), s can explain
   $\mathcal{F}_u(\ \neg errorMessage(\text{“}rm\ t^*\text{”})\ )$.
   Since $D =$
   $\{errorMessage(\text{“}rm\ t^*\text{”}) \leftarrow sideEffects(\text{“}rm\ t^*\text{”})\}$
   s can explain that
   $\mathcal{F}_u \cup D \models \neg sideEffects(\text{“}rm\ t^*\text{”})$.
2. s can consistently assume that D is consistent with the facts of u since it does not follow from the facts of s that u's facts imply the negation of D:
   $\neg(\ errorMessage(\text{“}rm\ t^*\text{”}) \leftarrow sideEffects(\text{“}rm\ t^*\text{”})\ )$.
3. By (23), s can explain
   $\Delta_u(\ errorMessage(C) \leftarrow sideEffects(C)\ )$
   for any C. Hence, s can explain that $D \subseteq \Delta_u$.

EXAMPLE 6.9. *Ascribing Belief*

Wilks, Ballim, and Bien (Wilks and Bien, 1983; Wilks and Ballim, 1987) propose various belief-ascription heuristics. One heuristic they propose is that the user believes the same propositions as the system does, provided that the proposition is a typical belief. This heuristic may have exceptions, so it is viewed as an assumption.

We can reason with such a heuristic in NT. Imagine that s believes that the world is round, and that it is typical to believe that the world is round. Then s assumes that u believes that the world is round. s has the following facts:

$$round(world) \tag{24}$$

$$typical(round(world)) \tag{25}$$

and the following default

$$\mathcal{F}_u(X) \leftarrow X \wedge typical(X) \tag{26}$$

From the above, s can explain that u explains $round(world)$. That is, s ascribes $round(world)$ as a fact of u, and since s has no conflicting facts in the model of u, u can explain $round(world)$.

If we add that s also has a fact that u has a fact that the world is flat (27), and that flat things are not round (28), then s cannot explain that u explains that the world is round.

$$\mathcal{F}_u(\ flat(world)\ ) \tag{27}$$

$$\mathcal{F}_u(\ \forall X\ \neg round(X) \leftarrow flat(X)\ ) \tag{28}$$

As in (Wilks and Ballim, 1987), this belief-ascription heuristic can operate on deeper levels to ascribe beliefs to agents that u is reasoning about. We can do this in NT by having as a fact of s that u has the heuristic as a default for all agents:

$$\Delta_u(\ \mathcal{F}_A(X) \leftarrow X \wedge typical(X)\ ) \tag{29}$$

s must also be able derive what is typical for u. One simple way to do this is for s to have a fact that the same things are typical for u as are typical for s:

$$\mathcal{F}_u(typical(X)) \leftarrow typical(X) \tag{30}$$

Note that, as with (Wilks and Ballim, 1987), defaults allow NT to build nested models as needed rather than having to precompute all levels. We do not need to specify in the KB all beliefs for all combinations of nested agents. These beliefs can be derived as necessary by using belief-ascription defaults.

7. Prioritized Nested Theorist

7.1. Conflict Between Levels

A technical problem with NT as defined above is that defaults on the metalevel are weak. If s has a fact that u has a default $p(X)$, then s can explain that u can explain $p(a)$ for an arbitrary object a. Now if s assumes that u has a fact $\neg p(a)$, an exception to the default $p(X)$, then s can explain that u can explain $\neg p(a)$. However, by definition 5.7, it also still follows that s can explain that u can explain $p(a)$.

EXAMPLE 7.10. *Multiple Explanations*

s has a fact that u has a default $p(X)$:

$$\Delta_u(p(X)) \tag{31}$$

and s has a default that u has a fact $\neg p(a)$:

$$\mathcal{F}_u(\neg p(a)) \tag{32}$$

Two mutually inconsistent explanations exist. In one, s assumes (32); hence s can explain $E_u^{\{\}}(\neg p(a))$. In the other, s assumes u's default (31) is consistent for $X = a$; hence s can explain $E_u^{\{p(a)\}}(p(a))$.

In example 7.10, statement (31) itself does not contradict statement (32). The default CAD, which s uses to assume that u's defaults are consistent, is in direct conflict with (31). For s to explain that u explains $p(a)$, s assumes

$$(\mathcal{F}_u \cup \{p(a)\} \text{ is consistent}) \leftarrow (\{p(a)\} \subseteq \Delta_u) \tag{33}$$

(33) and (32) are in direct conflict. Because of this conflict, s cannot block the assumption (31) of u by assuming that u has specific knowledge (32) of an exception.

7.2. Prioritized Defaults

Brewka has expanded Theorist to include defaults of different priority levels so that a default of higher priority can block a default of lower priority. This idea is more powerful than methods of removing multiple explanations (Poole, 1988) that do not allow one default to block another.

EXAMPLE 7.11. *Priority Levels*

Let D^i be the set of defaults of priority i, where a smaller i indicates higher priority. (1 is the highest priority level.)

$$\Delta^1 = \{\ \neg p(a)\ \} \tag{34}$$
$$\Delta^2 = \{\ p(X)\ \} \tag{35}$$

Given (34) and (35), defaults of different priority, $\neg p(a)$ can be explained, but $p(a)$ cannot.

We can use this idea to solve our multiple-explanations problem. By defining (32) to have priority over CAD, it is possible for (32) to block (33) so that only the desired explanation exists.

In general, we can solve this problem by having all defaults that are given by the knowledge designer, such as (32), to be of higher priority than CAD. The following is a non-technical summary of the definition of Prioritized Nested Theorist (PNT) that uses this idea.

DEFINITION 7.12. **PNT** *is a Prioritized Theorist system with a ML that refers to an OL (as described above) such that the metalevel Prioritized Theorist, s, has a fact that the user, u, forms explanations according to the definition of Prioritized Theorist. As in the definition of NT, s has a default that u's defaults are consistent. In PNT, this default is of lower priority than all other defaults of s.* [7]

8. PNT Examples

Examples 6.9 and 6.8 can be achieved in PNT just as they were in NT. Furthermore, example 7.10 works as desired in PNT. That is, s assumes (32), and concludes that u can explain $\neg p(a)$.

Some user models make use of reasoning with degrees of certainty (Rich, 1979; Chin, 1988, 1989). For example, when choosing which stereotype to use as a model of a user, there may be competing evidence, some stronger than the rest. Having prioritized defaults on the top level of reasoning enables this type of reasoning.

EXAMPLE 8.13. *Preferred Stereotypes*

Suppose that s has a default that u is a novice:

$$novice(u) \tag{36}$$

a higher priority default that u is a beginner if u has used a mundane command:

$$beginner(u) \leftarrow used_u(C) \wedge mundane(C) \tag{37}$$

and a fact that users cannot be both novices and beginners:

$$\neg(beginner(u) \wedge novice(u)) \tag{38}$$

With such defaults and facts, s assumes that u is a novice unless specific information about commands u has used suggests that u is more experienced, and should be classified as a beginner. This idea can be developed so that competing classifications are compared based on the strength of their preconditions. Defaults with strong preconditions can be given higher priority, so that the most appropriate classification can be found.

[7] There are other aspects of the definition of PNT that differ from NT. For example, in PNT, s can assume not only that u's defaults are consistent, but also that u's defaults satisfy priority constraints. That is, s can assume that u has no defaults of higher priority that contradict the given defaults. Details are presented in (van Arragon, 1990).

Sometimes it is useful to model the user as reasoning with degrees of certainty. Example 8.14 shows an example where s reasons about how u predicts what is true based on two sources of knowledge. The more reliable source of knowledge is believed. Again prioritized defaults, this time on the object level, provide a tool that enables this type of reasoning.

EXAMPLE 8.14. *Speech Acts*

A theory of speech acts specifies how utterances affect the beliefs of the hearer. An approach that can be implemented with PNT is to have the hearer u to assume that what the speaker s says is true:

$$\Delta_u^3(X \leftarrow declares_s(X)) \tag{39}$$

If we add to (39) that u has a fact that s declares p:

$$\mathcal{F}_u(declares_s(p)) \tag{40}$$

then u assumes that p is true. However, this assumption may be contradicted if u has a higher priority default that contradicts p:

$$\Delta_u^2(\neg p) \tag{41}$$

PNT provides a flexible language for stating the relative strengths of various defaults of u based on whether u believes s to be lying, or to be an authority, and so on. For example, u may assume at high priority that if agent A declares X, and X is within A's area of expertise, then X is true:

$$\Delta_u^1(X \leftarrow declares_A(X) \wedge expertise(A, X)) \tag{42}$$

Now if s declares p and is known by u to be an expert, u will assume that what s says is true.

With such defaults, s's model of u is automatically modified to take into account utterances of s. The change in u's belief due to utterances of s is built into the defaults and the definition of PNT. The knowledge designer's task is to define the contexts in which utterances occur, and to rank their priority based on which utterances are more likely to be believed by the hearer.

A full discussion of reasoning with uncertainty is beyond the scope of this paper. Prioritized defaults provide one way of doing this, and they are part of our tool anyway, since they are useful for solving the multiple explanation problem discussed in section 7.1. Only two levels of priority are necessary to solve this problem, but allowing an arbitrary number of priority levels gives rise to no new technical issues.

9. Limited Nested Theorist

So far we have modeled users as able to make assumptions to deal with incomplete information. An aspect of reasoning that we have overlooked is that a reasoner is limited in her ability to reason (Fagin and Halpern, 1988; Robert, 1988; Konolige, 1985; Levesque, 1984). We have shown that NT can model a user as being able to make unsound inference. Now we show how to model a user as being unable to make complete inference. (Notice that although we use logic, the tool we are proposing can represent both unsound inference and incomplete inference.)

NT's metalevel defaults are potentially a tool for reasoning about limitations. The metalevel can assume that in general an object-level agent can draw individual inference steps, unless the metalevel knows of specific constraints that prevent the agent from drawing an inference. Using defaults, s would assume that u can draw individual inferences. Using facts, s can reason about reasoning limitations that exist for particular agents in particular situations.[8]

This idea does not work in NT, because NT's metalevel does not reason about the inference itself. For the metalevel to make assumptions regarding individual inferences, the metalevel must represent the user's reasoning in greater detail. The metalevel must reason about, not only how the user's knowledge (facts and defaults) entail a goal, but also about the inference steps required to derive the goal. Given a representation of these steps, the metalevel can reason by default about whether the user is able to perform each individual step.

This approach is valid regardless of the underlying object-level reasoning procedure. For example, the underlying reasoning procedure itself need not be inherently incomplete. This is an advantage for a user modeling tool, because a tool should be flexible enough to model many different kinds of limitations. By having s reason about a reasoning procedure that is potentially complete, the tool need not be restricted to a particular class of limitations. Instead, the knowledge designer decides what type of limitations should exist in a specific application. The role of a user modeling tool is to provide an implementation of a flexible framework that allows the knowledge designer to specify limitations.

9.1. Modeling Linear Resolution

Although the design of a user modeling tool need not be committed to specific types of limitations, the design must be committed to a particular underlying

[8] (van Arragon, 1990) describes how to extend this so that s can use defaults to reason about reasoning limitations.

reasoning procedure. The tool must specify a particular object-level procedure and a metalanguage (ML) that can be used to specify limitations with respect to that procedure.

NT commits to modeling the user as forming Theorist explanations, but NT does not specify the underlying procedure to derive those explanations. The *implementation* of NT specifies the underlying procedure, but the implementation is orthogonal to the definition of NT.

Many considerations could go into choosing a particular reasoning procedure. The choice depends on psychological, philosophical, and practical dimensions. It is an empirical issue. To define LNT, this paper takes a simple practical approach. Rather than incorporate psychological models explicitly, we view LNT as a tool in which psychological models can be built. We define LNT using a simple logical reasoning procedure, and leave it up to the knowledge designer to build higher-level concepts. The advantage of this approach is that LNT is easy to build and to use. LNT is a good tool for further empirical study.

Our simple approach is to define u's reasoning procedure according to the details of the implementation of NT. Our implementation of NT is based on linear resolution. In this paper, we do not discuss in detail the implementation of NT, and hence we also present our definition of LNT without all details.

In LNT, s can reason about u's reasoning as a linear resolution proof tree. This idea is easy to implement, and is still sufficiently powerful for reasoning about many types of limitations. For example, s can reason about the size of the proof tree, about the length of proof branches, about whether particular predicates were used in the proof, and about whether individual proof steps are especially difficult. Various aspects of proof trees correspond to various types of resource limitations.

To reason about a user as performing linear resolution, we need to be able to refer to the steps of linear resolution in the ML. The key step of linear resolution as applied to Theorist is to chain on a fact.[9] We refer to this step using the ML predicate $infer_u(G \leftarrow B)$, where $G \leftarrow B$ is a fact of u.

As with NT, we define LNT by using facts and defaults of a Theorist system to define object level explanations. Instead of defining what is explained as what consistently follows from the facts and defaults, we define what is explained in terms of individual derivation steps from the facts and defaults. We take the simple approach of specifying chaining on facts as a default of s.

[9] Linear resolution is performed on clauses: statements of the form $G \leftarrow B$, where G is a literal and B is a conjunct of literals. Chaining on a fact is to prove that a goal G is true by showing that the goal B is true, where there exists a fact $G \leftarrow B$.

DEFINITION 9.15. **LNT** *is a Theorist system with a ML that refers to an OL such that the ML can represent the proof tree of object-level derivations. The metalevel Theorist, s, has facts and defaults that the user, u, forms explanations according to the definition of Theorist, such that the underlying reasoning procedure is specified as follows:*

1. *s has facts and defaults that specify that u reasons according to linear resolution (excepting the step of chaining on facts);*
2. *s has a default to assume that u can chain on u's facts. That is, s has a default "$infer_u(G \leftarrow B)$," meaning that u can infer G if u can infer B and has a fact $G \leftarrow B$.*

10. LNT Examples

To reason about limitations of u, s can reason using facts of the form

$$\neg infer_u(G \leftarrow B) \leftarrow \ldots \tag{43}$$

where the right side of (43) defines inferences that s believes cannot be made by u. To illustrate the capabilities of LNT, we consider examples using various kinds of limitations. These are adapted from other work regarding limited reasoning.

We can limit a derivation based on the number of steps it takes to derive a goal (Konolige, 1985). If s believes that u can draw only four inferences, it is inconsistent for s to assume that u has inferred more than four steps. Although a simplistic way of reasoning about limitations, this example illustrates the principle of using the metalevel to reason about a user's limitations.

EXAMPLE 10.16. *Length of Derivation*

s has facts that u has facts that imply p_n, such that the derivation requires n steps. [10]

$$\mathcal{F}_u(p_1 \leftarrow true) \tag{44}$$
$$\mathcal{F}_u(p_2 \leftarrow p_1) \tag{45}$$
$$\mathcal{F}_u(p_3 \leftarrow p_2) \tag{46}$$
$$\vdots$$
$$\mathcal{F}_u(p_n \leftarrow p_{n-1}) \tag{47}$$

s has a fact that u is unable to infer more than four steps:

$$\neg infer_u(G \leftarrow B) \leftarrow infer_u(G1 \leftarrow B1) \tag{48}$$

[10] $true$ denotes the empty conjunction. That is, $p \leftarrow true$ is the same as p.

$$\wedge infer_u(G2 \leftarrow B2)$$
$$\wedge infer_u(G3 \leftarrow B3)$$
$$\wedge infer_u(G4 \leftarrow B4)$$
$$\wedge G \neq G1 \ \wedge \ldots \wedge \ G3 \neq G4$$

The fact (48) prevents s from assuming more than four instances of $infer_u(G \leftarrow B)$. From these facts, s can explain $E_u(p_1)$ through $E_u(p_4)$, since they follow from the facts of u in less than five steps. However, s cannot explain $E_u(p_5)$. Although p_5 also follows from the facts of u, to explain $E_u(p_5)$, s would have to assume five instances of $infer_u(G \leftarrow B)$:

$$infer_u(p_1 \leftarrow true) \tag{49}$$
$$infer_u(p_2 \leftarrow p_1) \tag{50}$$
$$infer_u(p_3 \leftarrow p_2) \tag{51}$$
$$infer_u(p_4 \leftarrow p_3) \tag{52}$$
$$infer_u(p_5 \leftarrow p_4) \tag{53}$$

but these five assumptions together are inconsistent with s's fact that u is limited to four steps (48).

With this simple form of limited reasoning, we can model situations where u's facts are inconsistent, but u is too limited to realize the inconsistency.

EXAMPLE 10.17. *Undiscovered Inconsistency*
s has several facts regarding u's facts:

$$\mathcal{F}_u(p_1 \leftarrow true) \tag{54}$$
$$\mathcal{F}_u(\ p_2 \leftarrow p_1\ ) \tag{55}$$
$$\mathcal{F}_u(\ p_3 \leftarrow p_2\ ) \tag{56}$$
$$\mathcal{F}_u(\ p_4 \leftarrow p_3\ ) \tag{57}$$

s has a default that potentially makes u's facts inconsistent:

$$\mathcal{F}_u(\neg p_4 \leftarrow true) \tag{58}$$

However, s has a fact that u can reason using only two steps.

$$\neg infer_u(G \leftarrow B) \leftarrow infer_u(G1 \leftarrow B1) \tag{59}$$
$$\wedge infer_u(G2 \leftarrow B2)$$
$$\wedge G \neq G1 \ \wedge \ G \neq G2$$
$$\wedge G1 \neq G2$$

Therefore, in s's model, u is not able to detect the inconsistency. s can explain $E_u(p_1)$, $E_u(p_2)$, $E_u(\neg p_3)$, and $E_u(\neg p_4)$, since each of these only takes two inference steps for u. Without the reasoning limitation, u's facts are inconsistent, so nothing can be explained. (Explanations require facts to be consistent.)

Other reasons for lack of inference have nothing to do with limited resources. For example, lack of awareness regarding a goal may prevent someone from being able to derive the goal, even if the goal is a tautology (Fagin and Halpern, 1985).

EXAMPLE 10.18. *Awareness*

s has as facts that u is a hermit, and that hermits do not know of the existence of computers.

$$hermit(u) \tag{60}$$

$$unaware_A(computers) \leftarrow hermit(A) \tag{61}$$

s has a fact that an agent cannot infer a goal if the agent is unaware of some concept required to understand the goal.

$$\neg infer_A(G \leftarrow B) \leftarrow concept(G, C) \wedge unaware_A(C) \tag{62}$$

where concept(G,C) is true when C is a concept required to understand the goal G. For example, if G is the goal that the price of computers is rising ($rising(price(computers))$), s can explain

$$concept(rising(price(computers)), computers) \tag{63}$$

From these facts and defaults, s cannot explain that u explains the simple tautology that the price of computers is rising or the price of computers is not rising. That is, s cannot explain

$$E_u^D(\ rising(price(computers)) \vee \neg rising(price(computers))\ ). \tag{64}$$

Even though this tautology follows from u's facts and defaults (it follows from *any* facts and defaults), u, being a hermit, cannot derive the goal because u lacks awareness.

Other types of limitations can be represented using the same principles. (van Arragon, 1990) shows how Konolige's variation of the famous wise men puzzle can be implemented. In this example, an agent is limited simply

because a particular step of reasoning is difficult, and the agent is not smart enough to draw the inference. The constraint can be represented as follows:

$$\neg infer_A(G \leftarrow B) \leftarrow \mathit{difficult}(G \leftarrow B) \wedge \neg smart(A) \quad (65)$$

where inferences are classified on the basis of difficulty, and agents are classified on their inference ability.

Properties of LNT

From the above examples, we can see that several properties that are desirable for reasoning about limitations are achieved with LNT.

1 Explanations are not closed under implication. (Example 10.16)

2 Not all tautologies can be explained. (Example 10.18)

3 Explanations are not closed under valid implication. (See (van Arragon, 1990).)

4 Facts may be inconsistent without every sentence being in an explanation. (Example 10.17)

Furthermore, these properties are achieved with one simple concept. Since object-level inference is required to explain a goal, we can achieve these limitations by specifying object-level limitations on the metalevel. The limitations arise for a variety of factors, such as limited resource, lack of awareness, and others. We need not specify the limitations in advance, but may allow s to reason about them.

To use this technique the KB designer must decide what kinds of limitations are important, and include these as metalevel knowledge. LNT is intended to be a flexible tool with which to reason about many kinds of limitations. Therefore, limitations are considered as domain-dependent considerations, and are not built in to LNT.

11. Related Work

Reasoning about Defaults

Many user modeling systems employ defaults in some way. For example, the user modeling framework GUMS (Finin and Drager, 1986) uses defaults to build and maintain a model of a user's expertise. Different types of defaults are combined in GUMS, so GUMS's relationship with formal AI tools is unclear. In contrast, each of the nested levels of reasoning in NT is defined as a Theorist system. The formality makes the capabilities of NT clear, and enables augmented versions (PNT, LNT) to be defined. Another difference

is that GUMS does not represent a user's reasoning. It represents a user's expertise (similar to the *understands* predicate in example 3.2), but not individual beliefs.

Much user-modeling research employs formal tools that do not allow explicit representation of a user's default reasoning. For example, the underlying formalism of Wilks and Ballim (Wilks and Ballim, 1987) represents belief, but not inference. As such, it cannot be used to customize a model to a particular user's reasoning abilities as can NT (and especially LNT).

Another example is Perrault's (Perrault, 1988) theory of speech acts. The theory employs Reiter's default reasoning (Reiter, 1980). Perrault's augmentation of Reiter's default reasoning can reason about an agent's belief, and even an agent's reasoning, but cannot reason adequately about an agent's default reasoning.

Interestingly, Perrault's theory of speech acts requires a model of an agent's defaults. To reason about an agent's use of defaults, Perrault does not define defaults on multiple levels, but uses a default schema on the metalevel. This approach is not as expressive as NT, and thus is not as useful as a user modeling tool. The lack of expressability arises because it lacks the ability to express user defaults explicitly. For example, rather than expressing that $q \leftarrow p$ is a default of the user, it expresses a sort of approximation of this statement, such as "it is a default of the system that if the user believes p, then the user believes q." Default reasoning regarding the revision of a user's beliefs takes place solely on the metalevel, since the object level models only deduction, but lacks defaults. This works for some examples, but cannot deal with cases where the distinction between defaults on different levels is important.

A similar critique applies to the work of Konolige and Appelt (Appelt and Konolige, 1988), and of Joshi, Webber, and Weischedel (Joshi et al., 1984). Both systems reason about users (agents) that reason by default. However, neither explicitly represents defaults on different levels, and so neither is as expressive as NT.

In contrast, NT's reasoning about belief revision is a natural result of viewing a user as a default reasoner. Assuming that someone has a fact, and having a fact that someone uses a default are treated differently by NT. The distinction is lost by the systems mentioned above.

Reasoning Limitations

Syntactic approaches to reasoning about limitations base the definition of reasoning limitations on syntax. LNT is a syntactic approach because the metalevel reasons about object-level limitations based on the underlying reasoning procedure of the object level, which is based on syntax.

As Konolige argues (Konolige, 1986b), syntactic approaches may be more appropriate than semantic ones. Possible worlds logics, upon which most semantic approaches are based, are inappropriate in some ways for representing belief. Different possible worlds differ from epistemic alternatives that reasoning agents have. For example, a reasoning agent does not know the truth-value of complex statements (such as P=NP), whereas such statements are either true in all possible worlds or false in all possible worlds. This would mean that either an agent believes or disbelieves such statements depending on their truth-value. Avoiding this by modifying possible worlds often gives rise to counter-intuitive results, such as that incompleteness depends on inconsistency (Levesque, 1984), or that incompleteness is a combination of syntax and semantics (Fagin and Halpern, 1985).

Hadley (Hadley, 1986) suggests that the issue of differentiating beliefs should be based on a theory of intensions. Perhaps such a theory can be combined with a syntactic theory of reasoning limitations. The role of the intensional theory is to differentiate beliefs by defining equivalence classes. The role of the syntactic theory of reasoning limitations is to reason about how an agent can make inferences from one equivalence class to another.

The chief advantage of LNT is its flexibility in reasoning about limitations. Konolige's deductive model of belief (Konolige, 1986a) is similar to LNT in many respects. A difference is that limitations are defined procedurally in (Konolige, 1986a), whereas LNT permits defining limitations declaratively. A declarative specification of limitations may be an advantage for user modeling. Furthermore, in (Konolige, 1986a), the metalevel does not refer directly to the rules of inference, and therefore cannot reason about these. The limitations are fixed. In contrast, LNT can be used to reason about agents whose limitations are changeable.

Some work on logical omniscience has the goal of providing tractable KR tools (Levesque, 1984; Lakemeyer, 1987). The goal of LNT is to provide a flexible user modeling tool. LNT's implementation is efficient (based on resolution), but LNT is kept general so that inherently-difficult problems can be stated.

Achieving worst-case tractability comes at the cost of flexibility. For example, Levesque's explicit belief does not permit chaining as LNT does. Thus LNT can represent agents with more powerful reasoning (that is limited nevertheless). Furthermore, our implementation takes computational advantage of the defined limitations. The limitations specify when a proof can be terminated, enabling the implementation to avoid generating the whole proof tree.

A shortcoming of our approach is that we are restricted to using linear resolution as the underlying form of reasoning. Motivation for this is lacking,

except that it was easy to modify our existing code for NT to implement LNT. Our solution successfully defines a tool that allows a flexible specification of limitations, instead of being bound to using a fixed kind of limitation. However, to take full advantage of this approach, more work can be done to find other useful underlying reasoning methods.

12. Conclusions

NT is a formal tool that is a default reasoner that can reason about a default model of belief. NT is useful for user modeling as examples show: NT can build and maintain a model, and NT can reason about how a user reasons by default. Making this distinction between levels of reasoning, where each level can reason by default, provides a tool in which much of user modeling work can be incorporated. Furthermore, reasoning about how a user revises her default beliefs is more naturally accomplished.

The lesson we draw from this work is that user modeling research can make use of tools of formal AI, such as Theorist, but such tools may need to be augmented, as with NT. The goal of Theorist is an abstract version of a goal of user modeling: how to predict what is true in the world given incomplete knowledge. But Theorist itself does not specify a way to model an agent's reasoning, hence we augmented Theorist to form NT.

To permit s to assume exceptions to object-level defaults, we augmented NT to form PNT. PNT also allows the use of prioritized defaults for organizing knowledge. To permit reasoning about u's reasoning limitations, we formed LNT. LNT enables s to assume u is able to make an inference unless s knows of limitations on u's reasoning.

Implementation

NT has been implemented using a general technique to derive a nested reasoner from an interpreter written in Prolog (van Arragon, 1990). The technique uses metaprogramming. An interpreter for Theorist is converted into a Theorist meta-interpreter. This allows a Theorist interpreter to have as its facts and defaults a Theorist meta-interpreter. The Theorist interpreter corresponds to the reasoning of s, and the Theorist meta-interpreter corresponds to the reasoning of u.

The underlying implementation is based on linear resolution, and therefore is reasonably efficient. However, metaprogramming introduces overhead, since each step of inference on the object level requires several steps on the metalevel. van Arragon (1990) shows how this overhead can be removed using partial evaluation techniques. The overhead can be removed regardless

of how many levels of nested reasoning are present. However, this more efficient version is not fully explored or implemented. In any case, we can look forward to better performance in the future, since advances in logic programming research can be applied to NT.

Acknowledgements

Thanks for feedback from members of the LPAIG group at the University of Waterloo, especially David Poole, Randy Goebel, Robin Cohen, Scott Goodwin, Fahiem Bacchus, and Peter van Beek.

References

Appelt, Douglas E. and Konolige, Kurt: 1988, 'A Practical Nonmonotonic Theory for Reasoning About Apeech Acts'. In: *Proceedings of the Twenty-sixth Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, pp. 170–178.

Brewka, Gerhard: 1989, 'Preferred Subtheories: An Extended Logical Framework for Default Reasoning'. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1989*, pp. 1043–1048.

Chin, David N.: 1988, 'Exploiting User Expertise in Answer Expression'. In: *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, MN, pp. 756–760.

Chin, David N.: 1989, 'Knome: Modeling What the User Knows in UC'. In: Alfred Kobsa and Wolfgang Wahlster (eds.): *User Models in Dialog Systems*. Springer, Berlin–New York.

Cohen, Robin, Jones, Marlene, Sanmugasunderam, Amar, Spencer, Bruce and Dent, Lisa: 1989, 'Providing Responses Specific to a User's Goals and Background'. *The International Journal of Expert Systems: Research and Applications* **2**, 135–162.

Etherington, David W.: 1988, *Reasoning with Incomplete Information*. Pitman Research Notes in Artificial Intelligence, London: Pitman / San Mateo, CA: Morgan Kaufmann.

Fagin, Ronald and Halpern, Joseph Y.: 1985, 'Belief, Awareness, and Limited Reasoning: Preliminary Report'. In: *Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1985*, pp. 491–501.

Fagin, Ronald and Halpern, Joseph Y.: 1988, 'Belief, Awareness, and Limited Reasoning'. *Artificial Intelligence* **34**(1), 39–76.

Finin, Tim W. and Drager, David: 1986, 'GUMS1: A General User Modeling System'. In: *Proceedings of the Sixth Canadian Conference on Artificial Intelligence, 1986*, pp. 24–30.

Goebel, Randy: 1989, 'A Sketch of Analogy as Reasoning with Equality Hypotheses'. In: K. Jantke (ed.): *Analogical and Inductive Inference*, Volume 397 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, pp. 243–253.

Hadley, Robert F.: 1986, 'Fagin and Halpern on Logical Omnisciences: A Critique with an Alternative'. In: *Proceedings of the Sixth Canadian Conference on Artificial Intelligence, 1986*, pp. 49–56.

Hadley, Robert F.: 1988, 'Logical Omniscience, Semantics and Models of Belief'. *Computational Intelligence* **4**(1), 17–30.

Halpern, Joseph Y. and Moses, Yoram O.: 1985, 'A Guide to the Modal Logics of Knowledge and Belief'. In: *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 1985*.

Hayes, Patrick J.: 1977, 'In Defence of Logic'. In: *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 559–565.

Hoenkamp, Edward: 1987, 'An Analysis of Psychological Experiments on Non-monotonic Reasoning'. In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 1987*, pp. 115–117.

Jones, Marlene and Poole, David: 1985, 'An Expert System for Educational Diagnosis Based on Default Logic'. *Proceedings of the Fifth International Conference on Expert Systems and Their Applications*, pp. 673–683.

Joshi, Aravind K., Webber, Bonnie, and Weischedel, Ralph M.: 1984, 'Preventing False Inferences'. In: *Proceedings of the Tenth International Conference on Computational Linguistics*, Stanford, CA, pp. 134–138.

Konolige, Kurt: 1982, 'A First-order Formalisation of Knowledge and Action for a Multi-agent Planning System'. *Machine Intelligence* **10**, 41–72.

Konolige, Kurt: 1985, 'Belief and Incompleteness'. In: Jerry R. Hobbs and Robert C. Moore (eds.): *Formal Theories of the Commonsense World*, Ablex Publishing Corporation, Norwood, NJ, pp. 359–404.

Konolige, Kurt: 1986a, *A Deduction Model of Belief*. Pitman Research Notes in Artificial Intelligence, London: Pitman / San Mateo, CA: Morgan Kaufmann.

Konolige, Kurt: 1986b, 'What Awareness Isn't: A Sentential View of Implicit and Explicit Belief'. In: Joseph Y. Halpern (ed.): *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, Monterey, CA, pp. 241–250.

Lakemeyer, Gerhard: 1987, 'Tractable Meta-reasoning in Propositional Logics of Belief'. In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 1987*, pp. 402–408.

Levesque, Hector J.: 1984, 'A Logic of Implicit and Explicit Belief'. In: *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 198–202.

Levesque, Hector J. and Brachman, Ronald J.: 1985, 'A Fundamental Tradeoff in Knowledge Representation and Reasoning'. In: Ronald J. Brachman and Hector J. Levesque (eds.): *Readings in Knowledge Representation*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, pp. 42–70.

Perrault, C. Raymond: 1988, 'An Application of Default Logic to Speech Act Theory'. Technical Report CSLI-87-90, Center for the Study of Language and Information.

Poole, David: 1988, 'A Logical Framework for Default Reasoning'. *Artificial Intelligence* **36**(1), 27–47.

Poole, David, Goebel, Randy, and Aleliunas, Romas: 1987, 'Theorist: A Logical Reasoning System for Defaults and Diagnosis'. In: Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, New York: Springer, pp. 331–352.

Reiter, Raymond: 1980, 'A Logic for Default Reasoning'. *Artificial Intelligence* **13**, 81–132.

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

Shoham, Yoav and Moses, Yoram O.: 1989, 'Belief as Defeasible Knowledge'. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1989*, pp. 1168–1172.

van Arragon, Paul: 1990, 'Nested Default Reasoning for User Modeling'. Technical Report CS-90-25, University of Waterloo, Waterloo, Ontario, Canada.

Wilks, Yorick and Ballim, Afzal: 1987, 'Multiple Agents and the Heuristic Ascription of Belief'. In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 1987*, pp. 118–124.

Wilks, Yorick and Bien, J. S.: 1983, 'Beliefs, Points of View, and Multiple Environments'. *Cognitive Science* **7**, 95–119.