Combining power with tractability in belief models*

Jasper Taylor and Jean Carletta
Human Communication Research Centre
2 Buccleuch Place
Edinburgh EH8 9LW
United Kingdom
jaspert@cogsci.ed.ac.uk

Chris Mellish
Department of Artificial Intelligence
80 South Bridge
Edinburgh EH1 1HN

March 18, 1996

Abstract

Belief models are used in AI computer applications in a great many roles, both to keep track of the beliefs of users and to simulate the belief states of agents modelled by the system. Epistemic logic is often used as a paradigm for the behaviour of such models, but has problems of computational complexity, which are compounded by the need to include nonmonotonic inferences and change over time. Systems based on standard logics also sanction inferences which human reasoners would be unlikely to make.

In some aspects of reasoning, simplifications can be made which both reduce the computational problems and restrict inferences to those most plausible in humans. However, in order to reap the computational advantages of restricting inferences, a model-building approach similar to a compiler must be used. A description of the domain and the agents' knowledge is entered in a formal language, and translated into a restricted representation which allows a reasoning process embodying the restrictions and other efficiency features to be used.

The process of compilation and execution is described in detail, and compared with models which use different representations for different beliefs while remaining complete in terms of standard logic. Relationships between computational simplifications and improvements in psychological plausibility are discussed.

*The work described here is funded by Joint Councils Initiative UK grant #SPG9200139, running for three years from November 1992.

1 Introduction

As interactions between humans and computer systems become more widespread, complex and diverse, it becomes increasingly desirable for the designers of computer systems to take account of human reasoning in the design of user interfaces. In particular, interfaces to complex information-management systems need to tailor their behaviour to the expectations and deductive powers of the user, as well as interpret the user's requests in the context of what is known about the user. In order to address either of these problems they must represent the user's beliefs and inferences in a belief model, known as a *user model* in this role. Such models are also a vital component of intelligent tutoring systems, where a *student model* is used to vary the content and pace of tuition according to the student's perceived needs. Recently, research in agent architectures has started to make an appearance in systems with agent-based interfaces; here, belief models for the simulated agents are used to provide a basis for generating their behaviour, as opposed to the explanatory and predictive role of belief models for users.

A belief model in either case is merely one component of a model of rational action in general, which also requires some approach to representing goals, dispositions and plan generation and execution. Furthermore, an agent may acquire, modify or discard beliefs through many types of inferences, for instance induction, but the term "belief model" typically refers to a component that models changes that occur due to deduction from communication and other perceived events. Such belief models are generally framed in a propositional language, with some formal, symbol-based process of deduction. The language represents the objects of beliefs; these may themselves be the beliefs of other agents, which are referred to as nested beliefs. The other components of an agent architecture use the belief model as a database, adding representations of events to it and querying it about what is believed as a consequence of events to date.

There are three big problems with this approach to belief modelling. The first is the extensively studied problem of the computational complexity of logic-based descriptions of the world. To build belief models for richer domains requires more expressive logics; epistemic logics to cope with nested beliefs, non-monotonic or probabilistic logics to reason with uncertain inferences, and so forth. Secondly, there is the semantic complexity of trying to unify existing approaches to problems such as uncertainty and change over time. Only within the past few years have logicians started to produce logical formalisms incorporating model-theoretic semantics for scenarios combining multiple agents, change over time and uncertainty. Finally, reasoning systems based on standard logics do not have any inherent limitations to the complexity of problems they may attempt to solve in answering any query. This means they may come up with results which, while logically well-founded, are completely inaccessible to the human reasoning process they are supposed to represent.

Here we propose a belief model which alleviates these problems by using a representation of beliefs and a deductive process that inherently limits aspects of the complexity of inferences that can be drawn. Some of these limitations are implemented as restrictions on the domains that can be specified, these being:

- Logical implications in the domain must be representable as Horn clauses
- Times of beliefs cannot be specified

Others are the result of approximations built into the implementation of the model:

- Beliefs nested beyond a certain depth are treated as equivalent
- Negations at different belief nesting levels are not distinguished

The effect of these limitations is to make the reasoning process both more tractable and more humanlike in its limitations. While it is inevitable that such a restricted model will be unable to distinguish between some states of belief which might be afforded by events in the domain, we contend that this is not important in the domain of modelling human reasoning in the restricted context of interactions with an information system.

In order to illustrate the relation between the restricted model and the logical concepts which have generally informed the behaviour of belief models to date, we have designed an expressive language (called FEBL, Fairly Expressive Belief Language). This is a syntactically simplified version of Konolige's Hierarchical Autoepistemic logic (HAEL) ranging over expressions in a modal logic of belief. This language has a well-defined semantics and a concise syntax incorporating a sort hierarchy which is also used to express temporal regularities of the domain. To create a belief model for a given domain, we write a set of axioms in this language characterizing the behaviour of the domain and the knowledge of agents within it. These are then fed into a "compiler", which transforms them into an executable description of the state of belief of an agent in the domain, including that agent's beliefs about a dialogue partner, expressed in the restricted language. While some of the information in the FEBL description is lost in this process, the restricted model will never make any inferences that would be unsound in the original description, and its behaviour can be checked against that of a theorem prover in FEBL.

Section 2 will give an overview of the compiler by comparing it with other approaches to speeding up belief modelling, some involving restricting inferences and others using special representations but maintaining full expressiveness. Section 3 will describe the operation of the compiler in detail. Section 4 will go through the features of FEBL and the simplifications made during compilation, describing how each affects computational performance and psychological plausibility, and how some features address multiple issues in these respects. Section 5 will give a detailed description of the specification, compilation and execution of a task domain in which two agents have a dialogue.

2 Related work

Our compiler-based approach combines two tactics for improving that tractability of belief-modelling tasks. Firstly, it carries out simplifications which weaken the underlying logic. Secondly, it generates a representation which is customized for the rapid processing of the resulting model. We summarize related work according to which of these areas it addresses.

2.1 Systems embodying restricted reasoning

Several models of reasoning have been described which take some well-understood formal system and implement an analogous but simpler process. These divide into those which address the problem of tractability, and attempts to generate humanlike performance, although there is frequently overlap here. On the computational side, Selman and Kautz (1991) discuss possibilities for replacing a theory in general predicate calculus with one or more consisting exclusively of Horn clauses. In the general case there are many weaker Horn-only theories to choose from, and the way forward seems to be to use the system's "spare time" between queries to generate progressively better approximations. Our compiler forces the user to characterize the domain behaviour in terms of Horn clauses; correspondingly, restrictions are placed on the axioms of belief, a reified representation of which would otherwise require non-Horn formulae. It is not difficult to produce a version of our system capable of handling clauses in general CNF; Reiter and de Kleer (1987) characterizes deKleer's ATMS as an instance of a *Clause Maintenance System (CMS)* restricted to Horn clauses, and both this and deKleer's original work describe ways of generalizing its applicability.

Addressing the same problem, Lakemeyer (1987) also weakens the axioms of standard modal logics of belief in order that they can be added to a domain description in predicate logic without reducing its tractability. This accomplished at the expense of intuitive validity of the resulting system: it can, for instance, consistently represent "A believes (P and (A believes (not P)))".

2.2 Compilers and other accelerators

The ATMS (de Kleer 1986) can be viewed as a theorem-prover in a particularly complex multi-valued logic (see Ginsberg (1988, Sec. 16)), although it seems that the main reason for using such a logic is because the ATMS exists to support it. A more relevant characterization of the logical properties of the ATMS is Reiter and de Kleer (1987), where its function is described in terms of generation of clauses that can possibly be added to a theory in order to cause it to entail some conclusion. This is how it is used here, except such clauses are mapped onto sets of defaults in the logic whose tenability is then tested to ascertain the tenability of the queried conclusion.

An ATMS implementation forms a part of the compiled models generated by our system; the behaviour of the ATMS in general and in relation to its role here is described in subsection 3.3.1. For general reason-maintenance applications, Kelleher and van der Gaag (1993) added to the ATMS' advantages by putting an "up-to-date" marker on each node, so that when one is queried its label need only be recomputed if it cannot be shown to be unchanged since the previous query. Our compiled model is reminiscent of this approach in two ways: firstly, querying a term is what results in a corresponding node being created and justified, so there is no need to suppress the evaluation of labels for nodes once they are added. Secondly, in our model it is possible for a node's *justifications* to become out-of-date owing to change external to the reasoning system. So we use a labelling system similar to Kelleher's to indicate when a new node should be created for any given term, as described in subsection 4.3.2.

Etherington, Borgida, Brachman, and Kautz (1989) describe a less formal approach to implementing fast common sense reasoning, one with more emphasis on the desirable "humanlike" properties of such a system. It relies on the idea that a humanlike knowledge representation is tightly constrained by the number of models it supports; a database of givens will typically be augmented by closed-world assumptions which constrain the set of models, such that query-answering becomes more a matter of reading off properties of the few models remaining. While our rapid model does not explicitly constrain the models for its database, the ATMS does constitute a way of listing them explicitly, and the process by which questions are answered is similar.

An important point which is shared by our system is that the symbols for individuals in the rapid database stand in one-to-one correspondence with individuals in the domain of discourse, i.e., there are no quantified variables. They refer to a database of this sort as "vivid knowledge"; we, however, use the term "vivid" to describe the possession of knowledge implicit in an agent's familiarity with an individual or concept. This is the same sense as in Carletta (1992) but also differs from the sense in Kaplan (1969), where "vivid" describes an executable term (e.g., the value of an integer as opposed to something like "Sue's telephone number".

Pereira's process for generating fast approximations to complex grammars (Pereira and Wright 1991) embodied both simplification and compilation. It consisted of two stages, firstly the translation of an augmented phrase-structure grammar into an equivalent context-free grammar, and secondly the translation of the resulting CFG into a finite-state representation. While the second stage involves some degree of approximation, Pereira showed it to be exact for "interesting" subclasses of CFGs, and that the finite-state machine would in all cases accept any input which was acceptable to parsers based on the original grammar.

2.3 Hybrid reasoning systems

A knowledge representation language generally makes some distinction between terminological knowledge and assertional knowledge. The former relates to what categories of individual exist in the world and the sorts of relationships in which they can stand to each other. The latter describes the individuals and relationships that actually exist, and logical interdependencies between such relationships.

KL-ONE (Brachman and Schmolze 1985) was basically a rich terminological language with an assertional component added to take care of certain queries which did not fall within the competence of the terminological reasoner. While terminological logics are generally less expressive than assertional logics the subsumption relation in KL-ONE was still undecidable.

More recent approaches to logically complete knowledge bases, such as those described in (Frisch 1991), have taken advantage of the trade-off between the expressiveness of the terminological and assertional components, restricting the former to a sort hierarchy as in our own system, or providing some limited facilities allowing properties to be defined for individuals of given sorts.

2.3.1 Kobsa's BGP-MS

The user modelling shell system BGP-MS (Kobsa and Pohl 1995) is in some respects a descendent of KL-ONE. BGP-MS is a suite of packages for manipulating belief expressions, combined with a set of authoring tools enabling them to be tailored to particular application domains. The facilities for updating and querying the database in BGP-MS are suitably wide-ranging to reflect the different requirements of domains; for instance it can generate an "interview" to establish an initial model, or initiate a signal to its host application if it recognizes some important change that has taken place in the user's beliefs. Models generated by our compiler on the other hand are minimal in this respect; they merely accept updates and answer queries, and leave it to the application to decide which are the "important" beliefs that must be established and monitored. Any belief that is queried will be incorporated into the network representation of the rapid model, allowing subsequent queries about the same proposition to be answered very quickly.

The main methodological difference between our compiler and BGP-MS relates to the handling of complex belief expressions. BGP-MS uses one of three different formalisms to represent any given belief; these form a hierarchy of degrees of abstraction. At the lowest level there is SB-ONE, a language of terminological subsumption similar to that used in KL-ONE. Then there is BGDL (Belief and Goal Description Language); this consists of formulae in first-order logic divided between *views*, i.e., different chains of belief nestings. Finally there is an unrestricted language based on modal logic, which is used to represent relationships between specific contents of different views, and *modal-negated* formulae; those relating to the effects of absence of beliefs. Formulae presented to BGP-MS are assigned to the lowest level (least expressive) representation which can handle them; straightforward subsumption relations thus benefit from the rapid reasoning afforded by SB-ONE, while complex modal formulae are handled correctly if laboriously by a reification mechanism and first-order theorem prover.

Our compiler on the other hand distinguishes between only two levels of representation. Subsumption relationships are represented independently of belief spaces, and are fixed within a domain. All other belief expressions are trans-

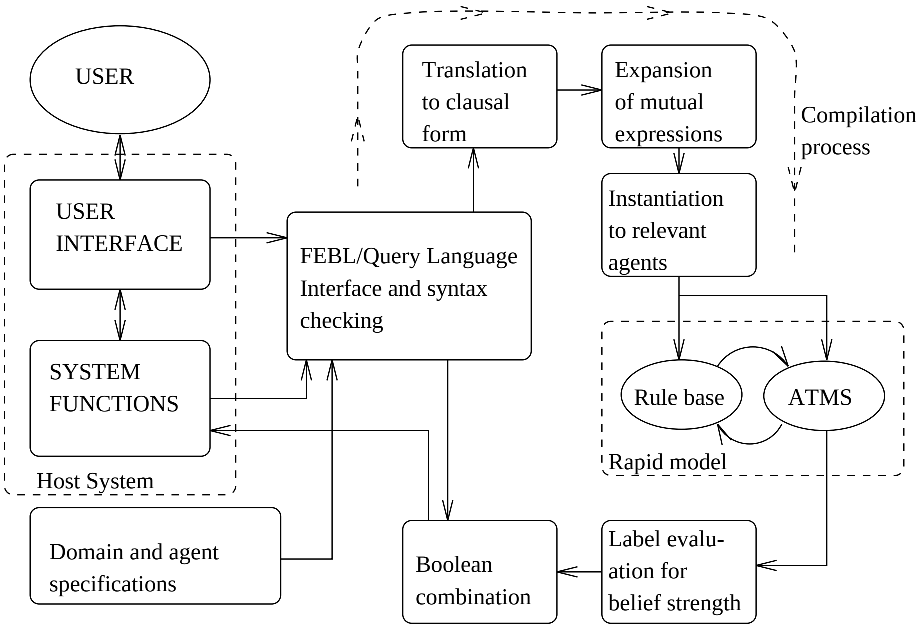

Figure 1: Interaction between the belief model and a host system

lated into a single formalism, which embodies belief nestings and negation only to a limited degree. Expressions in this formalism are divided into two groups whose representation varies depending on whether or not they relate to fixed individuals in the world. The salient point is that relationships that cannot be represented in the rapid model are weakened to produce simpler relationships that can, e.g., modal expressions containing multiple negations are replaced by nonnegated or singly-negated expressions which they subsume. No complex reasoning thus occurs in the execution phase of the model, which can thus be made fast and compact. We claim that the elimination from the model of the sort of reasoning that would require full theorem-proving capability does not adversely affect its performance in terms of generating humanlike belief updates.

3 Overview of the system

The main components of our system, their relationship with the host application, and the flow of information among them, are illustrated in figure 1. This section will describe the representation languages used at each point in this diagram, and the processes whereby these are translated and reasoned with at the appropriate points.

The figure illustrates the system's place in an overall package which uses the inference capability of the belief model to reason about the domain itself, as well as about the mental state of the user, whose beliefs are represented as nested beliefs of the system. There are two alternative configurations; as a belief model for a simulated agent, the box labelled "system functions" would contain only the functions appropriate to that agent, e.g., the planning of its own actions. The belief model could also be used as a user model independently of the beliefs of the system; this would be appropriate for a system carrying out inferences which a user would not be expected to follow, e.g., a very complex process control task. In such an application, the system's domain description would be separated from the domain and agent specifications in the user model; the system would be forced to "talk down" to the user in some respects.

3.1 The model specification language FEBL

The input language, FEBL (Fairly Expressive Belief Language) combines some of the features of a first-order logic of belief with a formalism for a nonmonotonic logic, namely Hierarchical Autoepistemic Logic (HAEL). We have added the notational features of a many-sorted logic (Walther 1987). These components were chosen because well-understood semantics and automated reasoning systems already exist for each of them, enabling checks to be made between the compiled model's behaviour and that specified by FEBL. It is anticipated that in the future, automated reasoning in integrated belief/temporal/modal logics such as Cohen and Levesque (1988) will be developed to the point where they could be used to specify belief models for compilation. A promising line of development in this direction is Ginsberg's (1988) framework for characterizing various logics which extend the expressivity of the first-order calculus, particularly since his characterization of the ATMS subsumes many of these logics and allows theorem provers in them to be implemented.

3.1.1 Adaptation of HAEL to form part of FEBL

HAEL (Konolige 1988) is a development of autoepistemic logic (Moore 1988). This is based on an object language, typically that of propositional or first-order logic, augmented by L, the autoepistemic operator. $L(P)$ can be deduced for each P that is implied by a theory, and $\neg L(P)$ for each P that is not. Defaults are thus expressed as implications involving $\neg L(P)$ where P is a condition that overrides the default. This language had the problem that if a theory allowed some P to be conditional on $\neg L(Q)$ where Q was conditional on $\neg L(P)$, it would have multiple consistent extensions; an example of such a theory is the *Nixon diamond:* "Republicans are normally hawks. Quakers are normally not hawks. Nixon is a republican and a Quaker." This yields two consistent extensions, in only one of which Nixon is a hawk.

Konolige described a way of overcoming this problem by specifying a hierarchy of strengths of evidence. All formulae in the database are assigned an evidential space, $\tau_{0,1,\ldots}$, depending on the strength of support for them. τ_0 is known as the "ground space" and contains no autoepistemic operators and hence no defaults. $\tau_{1,2,\ldots}$ contain propositions backed by progressively weaker evidence. The contents of each evidential space are propagated into all weaker spaces. The L operator is replaced with the L_n operator, which refers to the derivability of a proposition in evidential space τ_n. The L_n operator is only allowed in evidential spaces weaker than τ_n, thus eliminating circularities in default reasoning.

For instance, the proposition "If Tweety is a bird, and we have no stronger evidence that Tweety does not fly, then Tweety flies" would be represented as follows:

$$in\ \tau_n : bird(Tweety) \wedge \neg L_{n-1} \neg flies(Tweety) \supset flies(Tweety) \tag{1}$$

The L operator refers directly to the step in a reasoning process where conclusions in a higher space are to be tested.

In adapting the features of HAEL for the purposes of our representation language FEBL, we have replaced the L_n operator with the def operator:

$$def(n, \Theta) \equiv in\ \tau_n : \neg L_{n-1} \neg \Theta \supset \Theta \tag{2}$$

Furthermore, all propositions except for the autoepistemic formulae implicit in the use of the def operator are placed in the ground space. Any theory we might want to express in HAEL is equivalent to one which can be expressed within these restrictions using this notation.

The argument for this exclusive use of *single-place defaults* i.e., normal defaults with only one antecedent, is as follows:

- There is no call for nonnegated L_n operators. The values of autoepistemic formulae (those preceded by L_n operators) do not depend on the theory in the space in which they occur, so they cannot meaningfully occur as consequents of implications. Furthermore their semantics as antecedents is counterintuitive as, e.g., $L_n X \vdash Y$ would have the interpretation "Evidence in space n for X constitutes evidence in weaker spaces for Y (while weaker evidence for X says nothing about Y"). Such statements can be valid only if (a) Evidential spaces are given a probabilistic interpretation, which, while plausible, is specifically not our intention here, or (b) Weak evidence for Y is implicit in whatever might constitute strong evidence for X, a usage which should be discouraged as it goes against the recommendations of knowledge-engineering practitioners that the properties of knowledge should be explicit in its representation.

- Non-normal defaults, e.g., "In the absence of evidence that $\neg P$, conclude Q but not necessarily P", also seem to lack examples in real-world reasoning.

- Placing an axiom in a non-ground evidential space may give rise to inconsistency in that space while lower spaces are consistent. In such a case, replacing the axiom with a single-place default in the same space leaves the extension unchanged except for ensuring the inconsistency only occurs if the axiom is contradicted in the same space in which it occurs.

- Any normal default can be rewritten as an axiom and a single-place default, e.g., the Tweety example above becomes:

$$in\ \tau_n : def(n, normal(fly, Tweety)) \wedge$$
$$(bird(Tweety) \wedge normal(fly, Tweety) \supset flies(Tweety)) \qquad (3)$$

  with no change in the interpretations (the default $normal(fly, Tweety)$ should be read "Tweety is normal with respect to flying").

- Therefore, we can use a syntax that places all non-default statements in a single space, and uses the first argument of the def operator to indicate the space in which the default occurs, thus eliminating other explicit mention of evidential spaces.

3.1.2 Belief and autoepistemic operators

Moore's original interpretation of autoepistemic operators as referring to the system's ability to reason about its own knowledge was challenged by Shoham (Shoham 1988, p. 87); they are perhaps better regarded as referring purely to strengths of evidence. Thus there is no redundancy involved in their use in combination with the belief operator. Since both the knowledge operator of standard epistemic logic and the autoepistemic operator of HAEL are supposed to be applicable to any formula in the object language, we could allow them to be nested within each other; this would give the system the ability to distinguish between the attitudes "I believe with certainty level L that Fred believes P" and "I believe that (Fred believes P with certainty level L)".

However the complexity of any model-theoretic semantics of a language able to represent the latter proposition is such that we have excluded them, by specifying that belief operators can only be applied to formulae of the autoepistemic-operator-free subclass of FEBL. Konolige's intuitive interpretation of the evidential spaces in HAEL is that they refer to the strength of belief that the system is justified in holding in the propositions appearing there; we generalize the uncertainty indicated by the evidential space to apply to all agents involved in a belief. This appears to weaken the expressive power of the language; however, Jiang (1994) has shown that a formalism in which a multiagent epistemic logic forms the base language for an autoepistemic extension is equivalent to one where epistemic operators with agent subscripts extend an autoepistemic base language. Whether this result carries over from autoepistemic logic to HAEL is open to question, but it gives weight to the intuition that uncertainty about belief and belief about uncertainty can share a single logical representation.

A formal semantics for a language embodying this generalization appears in Taylor (1994b).

3.1.3 Sorts

We have incorporated a sort hierarchy into FEBL, in which the sorts of individual constants, the domains and arguments of functions and the arguments of propositions are encoded in a syntactically separate section. This is because:

1. Such a representation is more compact than the corresponding unsorted representation (Walther 1987).

2. Using sorts allows the syntax of expressions to be more tightly constrained, i.e., badly-sorted expressions can be rejected before compilation commences,

3. In theories representing evolving situations, e.g., dialogues, sort information about individuals may be carried implicitly along with other information, such as the occurrence of utterances. Using sorts allows the system to make the assumption that formulae derived from such perceptual information are well-sorted.

4. Defaults for categories in a subsumption hierarchy can be written without axioms or implications; e.g., "Anything that is a bird flies unless we know otherwise" can be written

$$\forall T : bird\ def(n, flies(T)) \tag{4}$$

rather than

$$\forall T def(n, normal(fly, T)) \wedge \forall T bird(T) \wedge normal(fly, T) \supset flies(T) \tag{5}$$

FEBL does not allow the "molecular" sorts (those defined in terms of boolean operations on atomic sorts, e.g., a mammal other than a dog) that appear in Frisch's general framework of sorted systems (Frisch 1991). However, as in Frisch's work, sorts are only used during unification, so there is no major obstacle to including molecular sorts.

Against these points must be weighed the disadvantages that a separate sort hierarchy does not allow agents to be represented as having false beliefs about sort relationships, e.g., "Calvin thinks bats are bugs", and that sort relationships or the sorts of individuals cannot change over time. This restriction could be detremental to the compiler's use in ITS applications, though we will illustrate later (subsection 4.2.1) how learning about the sorts of individuals could be modelled.

The many-sorted extensions of FEBL are preserved in the compiled model. This allows many problems to be represented in a greatly simplified form; Kobsa and Pohl (1995) discusses the advantages of representing subsumption relations as properties of sorts rather than expressions, and uses a system for translating relations expressed in FOPC into this form where possible. We have given our belief model the capability to draw inferences about the sorts of previously unknown individuals mentioned in its input; this capability, and the role of sorts in general, will be described in section 4.4.

3.2 Compilation step by step

We will illustrate the process of compilation and execution of a problem using a version of Konolige's example problem (Konolige 1988, p. 16) of deducing that an animal known to be a bat can fly. We shall present a more complex example involving nested beliefs in section 5. Konolige presents the bat example as follows:

Space	Formulae	Content
A_0	$B(a)$	A is a bat
A_1	$\forall x B(x) \supset M(x)$	All bats are mammals
	$L_0 B(a) \wedge \neg L_0 \neg F(a) \supset F(a)$	Bats normally fly
A_2	$L_1 M(a) \wedge \neg L_1 F(a) \supset \neg F(a)$	Mammals normally don't fly

This supports a straightforward inductive proof of $F(a)$.

To process this example in our system, we first add the terminological information that bats are mammals, mammals are individuals, any individual may or may not fly (and this aspect of individuals does not change over time), and Fred is a bat.

```
[individual [mammal [bat]]] -> _sorts;

[[fixed flies individual]
 [bat fred]] -> _decls;
```

(`fixed` is a propositional sort; these will be explained in subsection 4.3.2). Now we specify the axioms: we have translated the autoepistemic operators into the notation of single-place defaults described in subsection 3.1.2, removed the axioms subsumed by the sort hierarchy and generalized the default axioms so they refer to mutual beliefs about defaults that are likely to hold for all individuals of the given types, rather than just Fred. Variables with sort restrictions are written $V : sort$.

$$\begin{aligned} mb(\forall X : mammal\ def(2, \neg flies(X)))\wedge \\ mb(\forall X : bat\ def(1, flies(X))) \end{aligned} \tag{6}$$

Since the compiled model is restricted to the beliefs of just two agents, there is no need to provide agent-id arguments to the mb operator. They would have to be provided if the system was to work from a FEBL specification originally including shared beliefs between more than two agents.

3.2.1 Checking allowability of source

Currently, syntax checking is an aspect of compilation in general, with ill-formed expressions causing error messages and aborting compilation. To improve on this we are currently interfacing the compiler to an expectation-driven parser for context-free languages. This allows pseudo-natural language descriptions of domain rules to be used, after the terminology of the domain has been coded and used in conjunction with the domain-independent axioms of FEBL to set up the grammar. All such descriptions that parse correctly can then be translated into well-formed FEBL expressions.

As well as the syntactic restrictions on the input language, there are some constructs that do not have any unique equivalent in the compiled model. While some of these are ignored (such as distinctions between deeply-nested beliefs) others might correspond to regularities that could be expressed in a way that would allow them to be compiled, for instance relations between beliefs at non-adjacent levels of nesting. Such constructs are disallowed from the source, in order to indicate to the user when a concept might be expressed in a way more useful to the compiler.

3.2.2 Conversion into clausal form

This is a fairly standard operation, involving moving negations into conjunctions and disjunctions, distributing conjunction over disjunction, Skolemizing existential quantifiers and replacing universal quantifiers with implicitly quantified variables. The only non-standard operation is the handling of the propositional objects of belief operators (including default and mutual beliefs). These are converted to clausal form first, and the separate terms in each clause made into a new clause consisting of separate beliefs; the moving of negation does not violate soundness for Horn clauses, and the implicit quantifying-in involved in Skolemizing variables in belief contexts is dealt with in subsection 4.2. For instance,

$$bel(system, (A \wedge B \supset C)) \tag{7}$$

translates to

$$\neg bel(system, A) \vee \neg bel(system, B) \vee bel(system, C) \tag{8}$$

Only the explicit universal quantifiers in our example are affected by this step; subsequently, implicitly quantified variables with sort restrictions will be written $[sort\ V]$. So the example becomes:

$$def(2, mb(\neg flies([mammal\ X])))$$

$$def(1, mb(flies([bat\ X]))) \tag{9}$$

3.2.3 Expansion of mutual beliefs

At this point the database still contains a mixture of epistemic operators for individual and mutual beliefs, with no restrictions on the number and order of their nestings. In the rapid model we will conflate together all formulae with the same object-level proposition at two or more levels of belief nesting; this corresponds to three or more levels at this stage, as explicit mention of the system's viewpoint has yet to be removed.

To do this, we replace each conjunct containing an occurrence of the mb operator with four new ones, in which the mutual belief is substituted first with the object proposition, then with beliefs of separate universally-quantified variable agents at one, two and three levels of nesting. The def operators are moved to the outside of all belief operators, to preserve simplicity of semantics as described in subsection 3.1.2.

Sets of formulae which recursively imply more and more deeply-nested beliefs, such as

$$\forall A, B\; bel(A, prop) \supset bel(A, bel(B, prop)) \tag{10}$$

are disallowed in the source; such relations must be expressed using explicit reference to mutual belief, e.g.,

$$\forall A\; bel(A, prop) \supset bel(A, mb(prop)) \tag{11}$$

Since all axioms must be explicitly represented as being mutually known, this process generates all finite beliefs corresponding to mutual beliefs in the specification that will be relevant to the compiled model. Propositional variables in the formulae as they now appear can be treated as ranging only over object-level propositions. The bats example becomes:

$$def(2, \neg flies([mammal\ X])) \tag{12}$$

$$def(2, bel([agent\ A1], \neg flies([mammal\ X]))) \tag{13}$$

$$def(2, bel([agent\ A2], bel([agent\ A1], \neg flies([mammal\ X])))) \tag{14}$$

$$def(2, bel([agent\ A3], bel([agent\ A2], bel([agent\ A1], \neg flies([mammal\ X]))))) \tag{15}$$

$$def(1, flies([bat\ X])) \tag{16}$$

$$def(1, bel([agent\ A1], flies([bat\ X]))) \tag{17}$$

$$def(1, bel([agent\ A2], bel([agent\ A1], flies([bat\ X])))) \tag{18}$$

$$def(1, bel([agent\ A3], bel([agent\ A2], bel([agent\ A1], flies([bat\ X]))))) \tag{19}$$

3.2.4 Instantiation for a given pair of agents

We now have a description of the beliefs of of the universe of agents at the initial point of system operation. However, what we need is a belief model *from the modelled agent's point of view,* containing information only about the domain, and beliefs of the partner with respect to the domain and those of the modelled agent. The next stage therefore is to eliminate all assertions from the database except those forming part of this model. This conversion also implements one of the simplifications embodied by the rapid model — the elimination of deeply-nested beliefs. As described in (Taylor, Carletta, and Mellish 1996), private deeply-nested beliefs at the third level (e.g., Fred believes that Doris believes that Fred believes that Doris believes the door is open) and deeper need no explicit representation; instead, all beliefs at the second level and deeper are subsumed by *residual mutual belief (RMB)* in the topic.

To make this change, the database formulae are first rewritten as domain-level expressions preceded by lists of nested believers, e.g.,

$$\forall A, B, C \; bel(A, bel(B, bel(C, sunny\_today))) \tag{20}$$

becomes `[A B C] sunny_today`. Next, those where the believer list can be unified with an alternating list of `[self]` and `[partner]` are kept, and the others discarded. This is done because these are the only beliefs relevant to the rapid model's operation; only the modelled agent and its dialogue partner are considered, and self-nested beliefs (those which refer to an agent's beliefs about their own beliefs) are redundant as perfect introspection is assumed. So for the example above, A and C would unify with `[self]` and B with `[partner]`.

The belief prefixes thus generated are replaced with the type that are used in the rapid model: `[self]` becomes `[obj]` (signifying an object-level belief of the agent being modelled), `[self partner]` becomes `[bel]` (signifying a belief about a belief of the modelled agent's partner), and `[self partner self]` becomes `[rmb]` (signifying a residual mutual belief). Note that at this stage, any belief at the second level of nesting from the modelled agent's point of view is taken to denote a residual mutual belief. This is because private nested beliefs at that level (those that do not correspond to RMB's) are only generated (a) when reasoning about deception, or (b) from observations or reports of other agents' activities, (Taylor, Carletta, and Mellish 1996). When beliefs resulting from such activities are added to the model, they are marked as such so they can be placed in contradiction with existing RMBs (thus e.g., resulting in the removal of default RMBs) but do not lead to the establishment of RMBs themselves.

As all the beliefs generated by the expansion of mutual beliefs are universally quantified, this causes the final version of our example to appear as follows (we now use the list notation from the actual system since the contents of the database no longer correspond directly to formulae in epistemic logic):

```
[[def 2 obj - [flies [mammal X]]]]
[[def 2 bel - [flies [mammal X]]]]
[[def 2 rmb - [flies [mammal X]]]]

[[def 1 obj + [flies [bat X]]]]
[[def 1 bel + [flies [bat X]]]]
[[def 1 rmb + [flies [bat X]]]]
```

3.3 Running the compiled model

The compilation process has generated a list of Horn clauses, each literal comprising a belief-level indicator which is always a constant, followed by a formula which may contain individual, functional or propositional variables. Nonnegated literals may be defaults, in which case their whole clause represents a conditional default. This model is executed basically by carrying out resolution on these clauses, but rather than adding the results of resolution to the rule base, a reason maintenance system is used to keep track of the dependencies generated. Forwards and backwards chaining are used, and control of reasoning is effected by limiting inference to terms in which all individuals are instantiated to constants, and avoiding inference with clauses that have no justification.

3.3.1 Rule base and ATMS

The result of compilation of a domain description in FEBL is used to create a two part model, consisting of a rule-based system and a set of nodes and links in an ATMS (de Kleer 1986). Both of these components embody the same language; clauses of this language appear by default in the rule base, and in the ATMS if (a) they contain no variables and (b) all their terms can be made to hold by choosing some particular set of defaults from those represented. So far, in our example, all the clauses contain implicitly quantified variables so they are held in the rule base.

The ATMS is essentially a tool for nonmonotonic reasoning. It represents a set of propositions and the causal relations between them, and generates information on the sets of propositions which can be believed consistently. It does not deal with the semantics of propositional expressions; these are implicit in the causal relations specified to the ATMS.

Arbitrary propositions are assigned nodes in the ATMS, with designations as premises, assumptions or derived propositions. Premises are necessarily true, assumptions are true as long as they are consistent with whatever else is believed, and derived propositions are true iff implied by other propositions known to be true.

Interdependencies between propositions are represented by links between their ATMS nodes; a group of nodes can form a *justification* for another node, or a *contradiction*. The workings of the ATMS generate *labels* for each derived proposition – lists of the assumption sets which would entail their truth. New nodes and new interdependencies can be added to an existing ATMS network. The ATMS mechanism propagates any resulting changes in the labels along the links until once more they are consistent.

All ATMS nodes in the rapid model correspond to nonnegated literals. If a clause is to be represented in the ATMS, a node is created corresponding to each of its literals, and a justification for the originally nonnegated literal made from the conjunction of the originally negated ones. If there are no negated literals the node for the nonnegated literal is a premise, or an assumption (whose datum, or identity attribute, indicates the strength of the original default) if the clause was a normal default. If all its literals are negated, the clause is represented by a contradiction between their inverses. This table shows the relationship between Horn clauses and their ATMS representations.

Clausal form	ATMS form
`[[A]]`	A is a premise
`[[def n A]]`	A is an assumption
`[[- A] [- B] [C]]`	A and B together justify C
`[[- A] [- B]]`	A and B form a contradiction

3.3.2 Updates and queries

The interface to the model allows two types of interaction with the host system; updates and queries. Updates correspond to events occurring in the world in which the system operates; in a dialogue system these are typically utterances, either by the system itself or by the user. Updates could also communicate information to the belief model that has been generated internally by the host system; the belief model itself is intended to take care of all necessary deductive reasoning, but the host system might generate or retract rules through induction, and such changes would then need to be communicated to its belief model.

Queries allow the host system to obtain information from the belief model. A query is submitted by the host system in a query language which is a subset of FEBL, and the reply indicates the system's strength of belief in the queried proposition. The query language differs from the full FEBL in the following respects:

- It does not contain default operators.

There are no evidential spaces in the query language; the proposition is submitted and the model returns the identity of the strongest space in which it or its inverse is known to hold.

Reasoning represents the process of generating all derivable formulae for which (a) there are models, and (b) there is a chain of deduction connecting them to an expression which has been queried. As these are generated, they are represented in the rule base or ATMS as appropriate. In accordance with this specification, additions to the ATMS are triggered by the following circumstances:

- Addition of assertions in which either (a) there are no inverted literals, and no variables in the noninverted literals, or (b) all the inverted literals unify simultaneously with literals already represented by nodes in the ATMS and this unification causes all variables in the noninverted literal to be instantiated.

- Querying of a formula for which a backwards-chaining search on the rule base indicates an implication of it from a set of nodes already in the ATMS.

In either of these cases the new clauses will be added to the ATMS as described above, and may result in forward chaining. Where new nodes in the ATMS can unify with literals in the rule base, new rules are added with the corresponding variables appropriately instantiated.

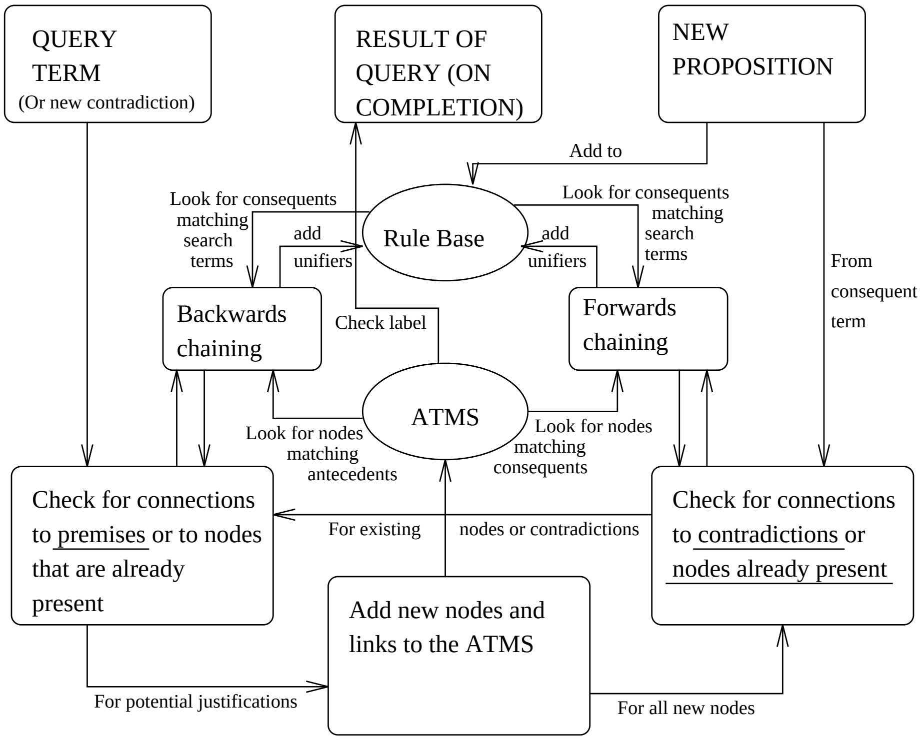

Figure 2: Interaction between the belief model and a host system

The process is illustrated in Figure 2. Once started, it continues until no further additions need to be made to the database; if it resulted from a query, then this is the point at which the result is computed as described in subsection 3.3.4.

Suppose we were to query the system resulting from the compilation of the bat example, to find out if the *user* believed that Fred could fly. The queried expression would appear in FEBL as:

$bel(system, bel(user, flies(fred)))$

This is compiled as described above to give

`[bel + [flies fred]]`

and a backwards chaining search of the rule base is initiated, i.e., we look for clauses that contain the queried literals with the same sign as that with which they appear in the query. This turns up the fifth rule in the list on page 15, whose only literal unifies with the queried literal. Since the result of this unification contains no variables, it is added to the ATMS, and as it has only one literal the ATMS representation is as a single node with intrinsic justification; since the rule is a default, this takes the form of an assumption.

The addition of this node initiates a forward chaining search, i.e., attempting to find clauses in the rule base which match one of the new literals but with the opposite sign. This turns up the second rule in the list. Again, unification

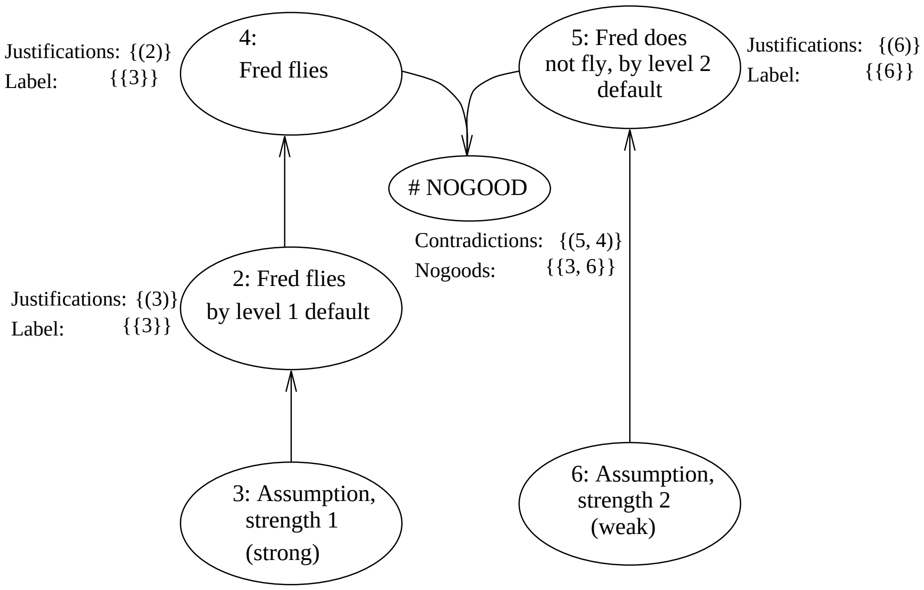

Figure 3: ATMS state for bat problem

instantiates all the variables, so this too is added to the ATMS as a justification, implying a contradictory state since it contains no nonnegated literals. However the basis for the contradiction is the queried node plus another assumed node (since it represents a default rule) so the first assumption is not removed. Figure 3 illustrates this state of the ATMS at this point as a network.

The query is then answered by a process involving examining the labels of the queried node and the nogood node, which will be described in subsection 3.3.4. Basically what happens is the strength of the assumption (`3`) underlying the queried proposition is compared with those with which it forms contradictions (`6`); since it is the stronger, it is also the strength of belief in the proposition.

A more complex example of reasoning in the rapid model will be presented in section 5.

3.3.3 Justifications and control of reasoning

A principal motivation for the development of the compiler has been to provide a reliable way of accelerating reasoning about beliefs in any given domain. This is achieved partly by the logical simplifications inherent in the process of compilation itself, and partly by restricting reasoning to relevant cases within the compiled model. The restrictions described here are the result of various compromises, and much remains to be done on the subject of getting the best performance out of the sort of model generated by the compilation process.

An important property of the model is that it separates the process of logical inference required to deduce which circumstances potentially lead to a given conclusion (modelled by the rule base) from the process of case-specific reasoning required to solve problems of conflicting defaults (modelled in the ATMS' internal state). Both of these processes are potentially intractable, but the complexity of the former process results from the availability of unrestricted inferences, a problem for which many potential solutions have been proposed (Levesque 1984; Halpern and Moses 1985). In the latter case, intractability is related to a profusion of alternative world models (Stenning 1991; Byrne 1991) and can be addressed by further altering the logical properties of the system to remove the less-favoured models as reasoning proceeds.

Here we have implemented a system which restricts logical deduction broadly to formulae which are true in some models of the system's nonmonotonic logic (not just the preferred model). Thus any improvement that can subsequently be made by a heuristic to reduce the space of models may also improve performance with respect to logical deduction. However, with the system as it stands, the need to deal with the non-preferred models leads to computational expense.

3.3.4 Extracting truth values from network state

The processes of clause search, unification and network addition described above result in the ATMS holding a representation of every Horn clause which (a) has no variables, (b) has at least one model, i.e., one consistent set of choices for the values of defaults in which the formula represents a truth, and (c) is derivable by alternating between forwards and backwards chaining from a clause which has been queried. It is necessary that the ATMS contain all such clauses, because any contradiction that is uncovered can affect the truth values of formulae arrived at by both forwards and backwards chaining from the contradicted pair.

Since the contradictions in the ATMS are recorded by marking the set of assumptions that can no longer be simultaneously held, the process of determining the strength of belief in a term can be carried out by an analogue of the recursive theorem-proving approach described by Konolige (Geissler and Konolige 1986), but without manipulating the actual formulae at all. Each literal corresponds to a node, which has a label consisting of a set of environments. Each environment is a set of assumptions from the conjunction of which the literal represented by the node follows.

The strength of belief in a literal is equal to the strongest of its labels. The strength of a label is equal to the weakest of its assumptions. The strength of an assumption is either its intrinsic strength (the first argument of the def operator of which it represents the truth, lower numerical values representing stronger assumptions) or nothing, if it appears in a nogood (a contradicted set) in which all the other assumptions are stronger than its intrinsic strength.

Evaluating the strength of an assumption has an effective procedure once the network has been built; if the intrinsic strength of the assumption is not the weakest in any contradiction in which it appears then that is its actual strength, otherwise the actual strength of the others must be evaluated. This will involve assumptions of successively greater intrinsic strength, and thus the process will end with evaluation in the strongest space.

4 What compilation achieves

Our purpose in designing an unrestricted language in which to describe potential domains and agents within them is not to capture all possible such domains, but rather to allow a rigid specification to be developed for a subset of the language which can be implemented in a straightforward and tractable manner. This specification corresponds to what can be implemented by the "compiled model" — a theorem prover working on a limited-abstraction language representation of the content of the FEBL model. Generating the compiled model involves not only rewriting the original input in a form that can be more readily used for automated theorem proving, but also implementing restrictions on what can be expressed in this form.

In what follows, we shall go through the features that make up the FEBL language as introduced in the preceding section, and with respect to each one describe the simplifications that are made during compilation, looking first at the computational motivation for carrying them out, then at what regularities exist in human information processing, or in domains of discourse in general, which might justify the inclusion of such simplifications.

4.1 Standard predicate calculus

While tableau-based theorem provers exist which can cope with representations that cover the full generality of first-order logic, many theorem-proving applications are restricted in scope to languages consisting of Horn clauses, i.e., conjunctive-normal form with no more than one non-negated literal in each clause. Such a restriction reduces the potential complexity of theorem proving, and also allows simple search algorithms (e.g., depth-first) to be used. Selman and Kautz (1991) discusses the possibility of using Horn clauses to approximate arbitrary theories in first-order logic; this is in fact a form of partial compilation, as the generation of Horn-only approximations to the current theory can take place in time that would otherwise be spent idle, for subsequent rapid query-answering.

This type of compilation is not available to our system as we describe it, since the Horn approximation is built into the translation itself. However, if we were confronted with a problem domain in which the initial Horn clause restriction proved significantly restrictive, it certainly would be possible to include an unrestricted representation, the FEBL theorem prover and a means of successively generating Horn approximations, so that idle time could be used constructively and queries not answerable from the original Horn approximation could be attempted, albeit with erratic response times.

More importantly, humans can have significantly more difficulty with problems that involve keeping track of disjunctions of possibilities, with some studies showing a relationship between problem-solving performance and the number of different models that must be considered to characterize the problem (Byrne 1991). A feature of problem-solving with Horn clauses is that disjunctions of non-negated literals are never derived. A mechanical theorem prover for a Horn clause theory need only keep a stack of the unresolved literals in any proof, whereas one for unrestricted clauses will need to keep a stack for each open branch.

As far as human reasoning is concerned, it is true that the maximum size of the stack of outstanding constraints to be integrated affects difficulty of problem solving (Carpenter, Miyake, and Just 1994) but this effect is less severe than the effect of increasing numbers of models. Both effects however seem to be more important than the lengths of chains of reasoning, which has also been limited in some attempts to restrict mechanical inference engines to the sorts of inferences humans make (Frisch 1987).

Another advantage of using Horn clauses is that they have an intuitive meaning along the lines of "If A and B and C, then D". This is part of what makes them useful in logic programming; other clauses are less easy to visualize. This principally affects people's ability to create representations of tasks, i.e., they are less likely to make mistakes if restricted to using Horn clauses.

4.2 Epistemic operators

There are a number of approaches to automated problem-solving in modal logics. The most general is reification (Moore 1985; Konolige 1986) but more recently there have been situation-based approaches (Appelt 1985; Lakemeyer 1987) and functional translation (Ohlbach 1991). Each of these techniques involves some form of translation which allows modal *theories*, which themselves describe domains, to be specified in a first-order metalanguage, thus allowing theorem-proving in modal languages which are otherwise difficult to use for reasoning. In addition, Ohlbach's method provides a way of handling implicitly quantified formula variables.

For our rapid language we have constrained the semantics of variables in such a way that no translation is needed in order to use what is essentially a first-order reasoning process. We will now discuss how we have dealt with the problems that normally prevent the use of first-order techniques, i.e.,

1. Identity across belief contexts. This characterizes problems that arise when an expression denotes different individuals for different agents, or two expressions denote the same individual for one agent and different individuals for another (the evening/morning star problem, Vardi 1983)

2. Knowledge that cannot be represented explicitly. An agent can know when another agent has information, e.g., a telephone number, without having that information themselves.

4.2.1 Identity across belief contexts

Problems of the first type above arise from attempts to characterize the behaviour of the equality relation over formulae containing belief operators using its standard first-order axioms. They typically involve agents being able to make unwarranted inferences about their worlds (e.g. Oedipus knowing he has married his mother). The most general alternative involves weakening equality to a notion of coreferentiality within worlds, by only including substitutivity axioms for function and predicate symbols. Since these axioms then have to be reasoned about as referring to nested beliefs, this is computationally very expensive.

However, we do not always need to implement a full-featured notion of equality. For instance, Shoham's very expressive temporal epistemic theory (Shoham 1988) made use of modal substitutivity axioms for expressions denoting times, thus assuming a model in which time expressions have the same interpretation for any agent. Furthermore, the idea that no expression denotes the same individual for two agents unless its extension turns out to be the same in all worlds accessible to those agents seems to place too tight a restriction on the identity of beliefs (Hewson 1994), so it seems appropriate to define some circumstances in which agents cannot get confused about identities.

This is provided by the compiler's use of a single sort hierarchy and sort declaration table. The implicit sort membership and subsumption predicates are common to all agents, i.e., agents can never be represented as being mistaken about relationships between sorts. They can be modelled as being mistaken about the sorts of individuals only by having them believe that all the references the system uses for one individual actually refer to another individual; e.g., to represent the system's beliefs that the sun is a star but that Fred thinks it is a moon, the system would categorize `sun-for-fred` as a moon, and have Fred believe as a default that all references to the sun were actually references to `sun-for-fred`.

In this case the process of enlightening Fred could be done by getting him to believe some proposition about what is called "the sun" that is incompatible with moondom, e.g., that it is bigger than the Earth. In the system's model, Fred's reaction to this will depend on the relative strength of his defaults. In one case, he would come to believe that "the sun" denotes `sun-for-fred`, which is an unusually large moon; the system must then convince him of some other attribute of the sun, or of moons in general, which does not hold for the other. Eventually Fred may come to believe that "the sun" denotes some new individual that is, e.g., too big to be a moon. He will keep his other beliefs about the referent of "the sun", except for it being a moon, which depended on it being equal to `sun-for-fred`. This process appears psychologically plausible as far as it goes, though some ITS applications might additionally require a representation of the process of adding to, or changing, a student's sort hierarchy.

Thus, by restricting the substitutivity axiom to apply only to formulae within the same belief context, and using the sort hierarchy to represent knowledge about individuals that we do want agents to be able to derive in every case, we achieve a compromise between expressiveness and tractability for our notion of identity.

4.2.2 Separating representation of knowledge from its content

The second problem mentioned above is in fact not confined to epistemic logic contexts, since even in classical logic we may need to reason about information that cannot be encoded in the formalism; for instance the location of a particular object may be needed to pick it up, but to reason about moving it at an abstract level does not require its actual coordinates to be represented. To handle such problems we allow the use of distinguished predicates which code for certain groups of propositions that need not actually be represented. The same notation is used to represent one agent knowing that another agent knows something, but not what it is, for instance a telephone number.

The classic example of the problem we are dealing with is the situation where it is known that John knows Sue's telephone number, but the number itself is not known. The formula:

$$\exists X knows(John, number\_of(Sue) = X) \tag{21}$$

fails to capture this knowledge as it is consistent with John not knowing Sue's number, but knowing that it is equal to something, e.g., itself.

In cases like this where the system does not know a proposition such as

$$number\_of(Sue) = \text{“}123\,456\,7890'' \tag{22}$$

but knows that an agent knows such a proposition to be true, we can use the *vivid* predicate and write

$$knows(John, vivid(number\_of(Sue))) \tag{23}$$

and from this we infer that John can phone Sue. Vividness of individual constants is an axiom, so an agent who knows an equality between any expression and an individual constant has the expression vivid. The special properties of the *vivid* predicate, i.e., that it is meaningless if outside a belief context or if negated, are incorporated into our compiled model by assigning the sort `virtual` to it in the system's sort hierarchy; other sorts applicable to predicates will be mentioned in section 4.3 in relation to the system's implicit temporal logic of belief.

Similarly we may need to represent a state in which an agent knows the truth value of a proposition when the system does not. This could be represented by writing

$$knows(John, P) \vee knows(John, \neg P) \tag{24}$$

but having introduced *vivid* as an indirect representation there is no reason not to introduce an operator to represent knowledge of whether a proposition holds:

$$knows(John, whether(P)) \tag{25}$$

This makes reasoning about such knowledge easier in a system based on Horn clause resolution, as it eliminates the disjunction. However, when using *whether*, it is generally inefficient to accompany every axiom which might be involved in deducing that an agent knows a proposition with another expressing that agent's knowledge whether the proposition or its negation holds; instead these should follow from the agent's belief in the proposition.

4.3 Temporal logic

A wide variety of formalisms have been created for representing the behaviour of domains which change over time; these range from axiomatizations of temporal relations in first-order logic (Moore 1985) through approaches that use modal operators (Cohen and Levesque 1987) to interval-based logics capable of representing large numbers of different ways in which the times of two events can be related (Allen and Ferguson 1994). As we are interested in domain-independent aspects of belief modelling, we have developed an implicit theory of belief change over time. For domains involving some simple changes over time, this theory can be extended to cover inanimate states of the world; for domains requiring more complex temporal logics, our model can only be used with a reified representation, and the temporal behaviour of beliefs treated separately from that of the domain.

4.3.1 Time relationships in the domain

For any application it is important to select a temporal logic which is expressive enough for the domain of interest but no more so, as both characterization of the domain and reasoning about it are more cumbersome with the exotic varieties. In the case of dialogue about an essentially static domain, a point-based representation such as the situation calculus (McDermott 1982) is adequate, as it can be assumed that the process of change to an agent's belief state resulting from one utterance will not continue past the next utterance. A slightly more expressive temporal logic involving a dense representation of time is sufficient for modelling the process of dialogue or discourse planning in such domains (see the modal truth criteria in TWEAK (Chapman 1987)). This representation can be reified with first-order logic plus a closed-world assumption (CWA) as its metalanguage. Thus it is suitable for encoding domains for the belief models generated by the compiler.

Taylor's CYNIC (Taylor 1994a) was based on a planning system which used a network-based representation of a simplification of Chapman's necessary truth criterion, and defeasible closed world assumptions. This was designed specifically for multiagent contexts where agents' knowledge about what actions would take place in a given time interval is continually updated; any justification for a belief about the state of the world at the end of some interval depends on:

- the state being asserted in the interval,
- any event known to happen in the interval not disturbing that state, and
- the assumption that there are no unforeseen occurrences in the interval.

When new occurrences are added to the reasoner's beliefs, the old CWA is defeated, the network representing this criterion is updated to include their possible effects, and a new CWA excluding them is made. CYNIC represented such beliefs about future states as being functions of time themselves; this gave rise to a lot of apparently unnecessary reasoning about persistence of beliefs.

4.3.2 Temporal properties of beliefs

The compiler described here embodies an important restriction with respect to reasoning about time: it will not represent beliefs at any time other than the present, at any degree of nesting. The domain-level content of the belief may include more temporal information if a reified temporal logic is being used, but no such information is stored in relation to beliefs themselves.

The basis for the psychological justification for this restriction has something in common with that for the avoidance of deeply-nested beliefs, namely people's unwillingness to reason with counterfactual beliefs. Considering what follows from the fact that Fred believes P when P is not the case can be avoided by attempting to inform Fred of the truth of the matter. Likewise, considering what follows from the fact that Fred believed P at time T is unlikely to be useful because if Fred has revised his belief that P then any beliefs he had in the consequences of P may also have been revised.

Computationally, the main benefit of this restriction is that all the effects of the system's domain-level beliefs at times other than the present are taken care of by the belief-revision procedure implemented in the ATMS. This allows the compiled model to represent the systems beliefs as they evolve over time, rather than recording their values throughout time. On the other hand, the values of domain-level propositions must be recorded throughout time, as their values at times other than the present are likely to be of consequence; this means that where they do change over time, beliefs about them must refer to their values at a particular time. Domain-level propositions that remain the same by default can be represented without time indices using *weak persistence*. This is an adaptation of the necessary truth criterion from CYNIC along similar lines to our representation of persistence for beliefs, where:

- The presence of an assertor is indicated by a justification for the proposition,
- The CWA is represented by an assumption at a weaker strength than any others in the model, and
- Any possible disturbance contradicting the proposition and thus overturning the weak persistence assumption.

Domain predicates whose values cannot change over the period being modelled, e.g., the colours of the blocks in a typical blocks-world problem, are assigned the sort `fixed`. Those that may change have the sort `mutable`, and persist only in conjunction with weak persistence assumptions; if they are subsequently contradicted, they are removed. `transient` predicates refer to things like utterances that do not persist at all.

Singly nested beliefs are a little more complex. They are taken as representing the system's current belief about another agent's current belief. However, unlike a domain-level proposition referring to a particular time, the other agent's belief may change and the system's belief about it thus need to be updated. The mechanism for this is fairly simple; whenever a belief revision is completed which leads to a new justification for another agent's belief, or a new justification following from it, the belief is represented by a new proposition which is implied by the old one. Thus if the system deduces that an agent no longer has a certain belief, then some assumptions which led to any earlier conclusion that the agent did have that belief must be removed; it does not matter whether they were the system's own, or relating to the agent's other beliefs. On the other hand, if the belief is reinstated, those assumptions remain contradicted. Predicates behaving like this are assigned the sort `monotonic`.

The other category of belief represented by the system is residual mutual belief (RMB), the combined representation of beliefs in a proposition at levels of nesting deeper than the first. Since they include beliefs about first-level nested beliefs, which can themselves acquire new justifications, it seems plausible that they should require the same system for updating. However the ways in which RMBs are established and persist in cooperative dialogue are limited in such a manner that these beliefs can be treated the same way as domain-level beliefs, i.e., declared `fixed` and left to the ATMS's belief revision mechanism.

4.4 Many-sorted notation

The advantages in efficiency of using a many-sorted notation have already been explained in subsection 3.1.3. As well as these gains in notational efficiency, there are computational advantages which follow from the limitations on variability of sort relationships between the different nested belief spaces in the system. We have to code explicitly the individuals whose sort might be a matter of inter-agent confusion as described in subsection 4.2.1, which allows sort relationships in the general case to be common to all agents, thus saving time overall in situations where most sort relationships are unambiguous. It may be found in some ITS applications that confusions about the sorts of individuals and even about the sort hierarchy are so commonplace that the best solution is to revert to a predicate representation of some sort properties.

The compilation process as it stands is not directly relevant to these issues; they apply to any belief-modelling formalism. However, an extension to the compiler could be made to allow the domain specification to be entered in an unsorted language. The implications between unary predicates in this specification could be generated as part of the compilation process and used to build up a sort hierarchy as described in Frisch (1991) for the rapid model. Similarly, domain descriptions containing extensions in simple temporal logics such as the situation calculus could be translated into the implicit temporal representation of the rapid model, if they conformed to certain restrictions e.g., no implications between sentences referring to non-adjacent points in time.

Furthermore, we have already described in subsection 3.2.1 how the sort information supplied for the model can be used as the basis for a context-free grammar allowing the domain specification to be entered in pseudo-English. It would be possible to create a similar grammar allowing the sort information itself to be entered in the same way. The process of domain specification entry would then continue with the compilation of this information to produce the grammar for the assertional part of the specification.

4.5 Prioritized defaults

Konolige (Konolige 1988, p. 23) describes a resolution theorem-proving system for HAEL, which can be built around a theorem prover for the object language (the HAEL language without the L operators: first-order logic in Konolige's examples). The basic idea is that the theorem prover runs in the evidential space of the queried formula, and whenever $\neg L_n P$ has to be proved, an attempt is made to prove P in space n, the original proof succeeding if the new one fails.

While this technique could be applied to FEBL with or without compilation as described here, it is clear that reasoning is likely to be duplicated in different evidential spaces. The level of duplication will depend on the number of evidential spaces (though any more than one will result in some) and the prevalence of defaults. However, just as using a single-environment reason maintenance system avoids duplication of reasoning operations in autoepistemic logic (Moore 1988; McAllester 1980; Konolige 1986) the multiple-environment ATMS stores interdependency of formulae independently of the presence or strength of evidence for each of them.

In response to a query, our compiled model will first create ATMS nodes for propositions which have justifications in any evidential space. These nodes are linked into a network according to the rules of inference and the specification of the language; this network embodies all the syntactic relationships between the formulae in the model. Assumptions in any space are propagated simultaneously through the network, in an operation that is computationally inexpensive compared with deriving the interdependencies in the first place. When this has been done, the evidential spaces in which propositions are believed can be derived by the procedure described in subsection 3.3.4.

The compiled model thus implements the FEBL specification in full with regard to prioritized defaults, with a significant speed advantage over a theorem-prover-based implementation.

4.6 Closed-world assumption

The principle that only propositions relating to identifiable individuals are represented in the ATMS has the consequence that justifications within the ATMS cover only the Herbrand Universe of the problem. Since all paths that could lead to the creation of justifications are explored, the justifications actually generated completely cover those available from Herbrand universe sets.

As a result, the system can be augmented to include evidential reasoning by agents, for instance if either A or B implies C, and the agent observes that C is the case, he concludes that either A or B. Such conclusions require a closed-world assumption (CWA), which could be either implicit, or explicit and defeasible in the event of new individuals being added to the universe by new observations.

We have carried out experiments in which this principle is applied by agents reasoning about other agents' beliefs; an example of a rule used in such a system is shown in section 5.2.1. However, the representation of such inferences required that the rapid model was able to represent non-Horn clauses, which entailed problems in the control of reasoning. Therefore the currently complete version of the compiler does not implement such deductions.

Psychologically the use of the CWA in problem-solving corresponds to the expectation in dialogue that all relevant information has been given, as represented by Grice's maxim of quality. Observance of Grice's maxims (Grice 1975) is a requisite of cooperative dialogue. Violations, such as to ask a partner for a solution to the Missionaries and Cannibals problem (which involves crossing a river in a complex series of moves) and then rejecting the answer in favour of "They cross this bridge I didn't mention before", are clearly non-cooperative.

5 Worked examples

The original domain in which we studied the role of nested beliefs in communication was a model of agents carrying out the navigational task in the HCRC Map Task Corpus (Anderson et al. 1991). This is a simple navigation problem where two people each have a different map of a region, and one (the instruction giver or IG, typically "she") guides another (the instruction follower or IF, typically "he") along a route marked only on her map by referring to landmarks they might encounter along the way. The task is complicated by the possibility that not all the landmarks appear on both maps (see Figure 4 for an example). A description of a domain-specific belief model for agents carrying out this task appears in (Carletta 1992); the domain, however, can readily be made more or less complex than the version described there, by selectively including complicating features present in the original study, such as duplicated or differently-named features on the maps.

5.1 Establishing terminology for the domain

The first stage in creating an implementation of an agent model for this task was to decide what knowledge needed to be represented, by working backwards from the route follower's goal of being able to mark the whole route on his map as accurately as possible. This is imagined as constituting knowledge of a series of locations, corresponding to the parts of the route at which descriptions of its position are given.

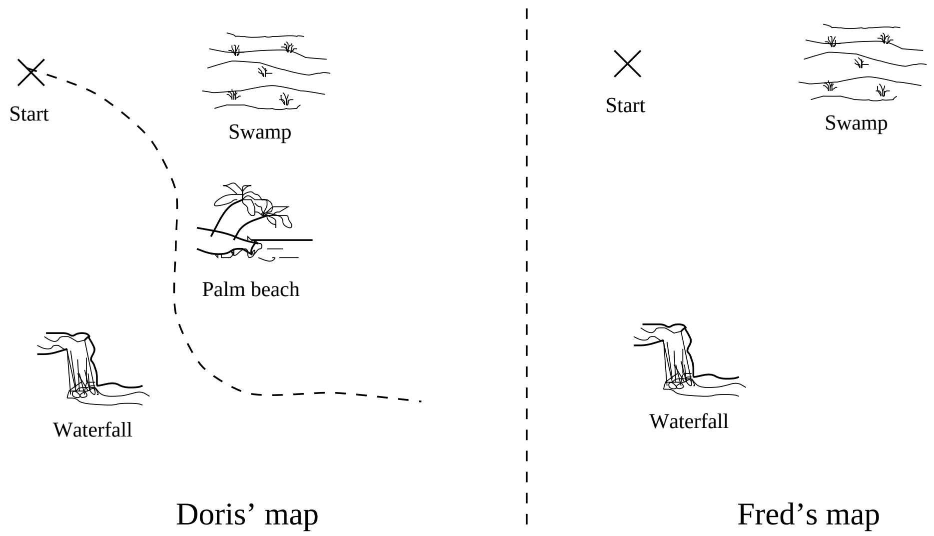

Figure 4: Part of a typical pair of maps in the Map Task

Because agents need to represent other agents' knowledge of locations they do not share, a *vivid* predicate is introduced for them. The roles of descriptions in providing information about one location in terms of others can be expressed in terms of relations between the vividness of locations. Indeed, having characterized behaviour of vividness thus, we can omit referring to any reasoning done by agents about actual positions represented in a coordinate system; if a dialogue results in an agent believing $vivid(swamp)$, it follows that had we included the corresponding axioms for reasoning about coordinates, the agent would know the coordinates of $swamp$.

5.1.1 The Sort Hierarchy

The terminological hierarchy is established by examining a freehand description of the rules of the interaction, though possible alternatives to this step are discussed in subsection 4.4. The following hierarchy is used:

```
[individual  [proposition  [virtual]
                           [transient  [mutable
                                          [monotonic  [fixed]]
                                          [contingent [fixed]]]]]
             [relation     [relation_1]
                           [relation_2]]
             [location     [description]
                           [map_item   [landmark]
                                       [route_section]]]
             [agent        [route_giver]
                           [route_follower]]
             [action       [utterance]]
             [integer]]
```

`proposition` and its subsorts are used to restrict the roles that propositions can play when reasoning about beliefs. `virtual` describes a proposition which is not directly expressible in a declarative utterance, typically the fact of having something vivid, as described in subsection 4.2.2. The other subsorts represent the relationship between propositions whose values change over time according to different constraints, as described in subsection 4.3.2.

`relation_1` and `relation_2` cover one- and two-place preposition functions such as $left\_of$ and $between$.

A `location` is a point on the map. Descriptions have been conflated with the points to which they refer, and map items (anything appearing on either agent's map) have been conflated with their positions; these merely make the representation more compact. If we are later to expand the task to include agents' reasoning with the content of the landmark representations, e.g., to note the physical similarity between a "swamp" and a "marsh", then we will need to model landmarks as individuals of which the location is a property, rather than simply as special types of location. Route sections are differentiated from landmarks because of the different assumptions each agent makes about their presence on the other's map. Similarly, `agents` are differentiated into route givers and followers to enable these different assumptions to be modelled.

The only `actions` in this domain are `utterances`; although in the experiment the route follower was expected to draw the route on his map, the model does not distinguish drawing the route from knowing it. `integers` are included because they are recognized by the system as specifying assumption strengths, but like the subsorts of `proposition` are of no interest to the model builder in this domain.

5.1.2 Sort declarations for terms

During the operation of the compiled model, forwards and backwards chaining is done by matching terms in the network representation with terms in the rule base which contain implicitly quantified variables. The unification of these terms is carried out with reference to the sort hierarchy in order to generate only well-sorted unifiers, so the sort restrictions are essential to ensure sound inferences. Sort restrictions are also used to determine the sorts of new individuals when the system acquires knowledge by way of domain events; for instance, when modelling a route follower who hears the utterance "The first section goes left of the swamp", the individual referred to as "the swamp" is constrained to be a landmark by its place in the corresponding proposition.

The sort declarations for individual constants, functions and predicates used in the map task simulation are as follows. The first item in each list is the sort of the term, followed by the name of the term and the sorts of any arguments. The only distinction between predicates and functions is that the sorts of the former are subsorts of `proposition`.

```
[monotonic bel agent monotonic]
[monotonic bel agent virtual]
[mutable bel agent mutable]

[monotonic mk monotonic]

[virtual def integer proposition]

[fixed - fixed]
[monotonic - contingent]
[contingent - monotonic]
[mutable - mutable]

[virtual vivid location]

[virtual mapped map_item]

[fixed desc map_item description]

[description to_1 relation_1 map_item]
[description to_2 relation_2 map_item map_item]

[transient do agent action]

[utterance assert fixed]
[utterance wh_ask virtual]
...

[route_section section_1]
```

```
...

[landmark palm_beach_1]
...

[route_follower fred]
[route_giver doris]

[relation_1 left_of]
...

[relation_2 between]
```

The declarations of `bel`, `mk`, `def` and `-` are common to all domains. `vivid` always returns a `virtual`, and here it is declared to be applicable only to locations (and thus to map items); if the task required agents to represent each other as knowing whether propositions held, then the "whether" operator would also return a `virtual` proposition. `mapped` is another `virtual`, but whereas `vivid` applies to locations and represents knowledge of a position, `mapped` applies only to map items and describes what an agent knows by virtue of having the item actually on the map in front of her.

As indicated in the type hierarchy, `descriptions` are only relevant in the sense that they specify `locations`; they can be true or false of `map_items` according to whether the locations match. A description itself consists of a one- or two-place relation with a map item for each place; alternatively we could allow descriptions to refer to locations rather than map items, allowing descriptions such as "between the palm beach and a point left of the swamp".

Since all `actions` are `utterances` in this domain there is no need to make them separate sorts apart from that of clarity. The eight utterance functions correspond to a selection of the utterance "moves" in the Map Task simulation described in (Carletta 1992); table 1 gives examples of their use. Note that the proposition to which an utterance refers may not be mentioned in the utterance if it is not an initiation.

The rest of the declaration section merely lists the sorts of the individuals about which one or both the agents know at the start of the run.

5.2 Specifying the domain axioms

Characterizing the domain appears to involve creating the assertional and terminological components of the domain simultaneously; however, as a final check the terminological component can be built into a context-free grammar and used to parse a pseudo-English representation of the assertional component. The grammar corresponding to the terminology described above, and an example of a parse tree for the pseudo-English representation of one of the axioms described here, is presented in appendix B.

Utterance type	Role	Example
assert	initiate/reply	"The first section goes below the swamp."
wh-ask	initiate	"Where is the swamp?"
yn-ask	initiate	"Do you have the swamp?"
yes	reply	"Yes."
no	reply	"No."
done	initiate	"I have the swamp."
dont-understand	reply	"I do not understand."
reply-ok	reply	"OK".

Table 1: Utterance types, roles and examples

The axioms characterizing our domain split into those relating to the beliefs that are brought about by hearing utterances, those relating to inferences about the domain, and those describing what the participants initially believe: the contents of the maps in front of them, and the facts about the task as related to them by the experimenter.

5.2.1 The effects of utterances

Part of the design of the task is that the two participants can hear each other, so all utterances are co-presence events and hence mutually known to have occurred. This can be expressed directly as follows:

$$\forall X\!:\!agent\; Y\!:\!agent\; V\!:\!utterance\; mk(do(X,V) \supset bel(Y, do(X,V))) \tag{26}$$

Expansion of the mk operator during compilation of this term will produce a series of axioms expressing how each nesting level of belief in an utterance follows from the previous one.

There is an axiom describing the effects of each utterance type in terms of beliefs in its proposition. Here there are two possible ways of coding the effects of utterances; they can either be represented as following from the utterance itself, or the belief state of the speaker predicated by the utterance can be represented as leading to the utterance, allowing the addressee to infer it through the Closed World Assumption (CWA) as described in subsection 4.6.

For instance, the following axiom expresses the fact that an agent who asks where a map item is does not have its location vivid:

$$\forall X\!:\!agent\; Z\!:\!map\_item\; mk(do(X, wh\_ask(vivid(Z))) \supset \neg bel(X, vivid(Z))) \tag{27}$$

whereas the CWA version expresses the fact that an agent who doesn't know where a map item is will, under certain unspecified circumstances, ask where it is:

$$\forall X\!:\!agent\; Z\!:\!map\_item\; \exists P\!:\!virtual$$
$$mk(\neg bel(X, vivid(Z) \wedge P) \supset do(X, wh\_ask(vivid(Z)))) \tag{28}$$

Compilation of the first version is straightforward; in the second version, `P` is Skolemized into a function of the agent and map item. When reasoning with the CWA, it follows from the occurrence of the utterance that the speaker doesn't have the map item vivid, and that `P(X, Z)` although no further reasoning can be carried out with this expression. Although the CWA version is the more detailed representation of utterance effects, the current version of the compiler does not support it.

5.2.2 Inferences about the Domain

These simply specify under what circumstances an agent can come to have a location vivid. There are several choices to be made here; some we have made already by specifying that map items and descriptions are the only entities whose types are subsumed by "location", and thus the only ones to which "vivid" can apply. The axioms we actually use allow vividness of a map item to be inferred from a "describes" proposition relating it to a vivid description. Vividness of a description in turn is inferred from the conjunction of the vividnesses of all the map items to which it refers:

$$\forall X : map\_item\ mk(mapped(X) \supset vivid(X)) \tag{29}$$

$$\forall X : map\_item\ Z : description\ mk(vivid(Z) \wedge desc(X, Z) \supset vivid(X)) \tag{30}$$

$$\forall X : map\_item\ R : relation\_1\ mk(vivid(X) \supset vivid(to\_1(R, X))) \tag{31}$$

$$\forall X : map\_item\ Y : map\_item\ R : relation\_2$$

$$mk(vivid(X) \wedge vivid(Y) \supset vivid(to\_2(R, X, Y))) \tag{32}$$

An alternative to this would be to define an identity relation, which could then be used to specify that the locations of items were the same as those of their descriptions; the second axiom above could then be replaced with

$$\forall X : map\_item\ Z : description\ mk(desc(X, Z) \supset (X = Z)) \tag{33}$$

Similarly the relationship between vividness of the objects and the referents of single-place descriptions could be made bidirectional, allowing agents to infer the position of the referent if they already know that of the item described. Here the motivation for limiting the inferences of vividness is that people often fail to make such reverse inferences.

5.2.3 Rules of the task

The information available to each agent from their map is expressed as a series of beliefs about `mapped` properties, plus `desc` relations for all sets of landmarks close enough to be used in descriptions of each other. We then require axioms for the initial shared knowledge that the route giver has the whole route and the follower has none of it:

$$\forall X : route\_section\ Y : route\_giver\ mk(bel(Y, mapped(X))) \tag{34}$$

$$\forall X : route\_section\ Y : route\_follower\ mk(\neg bel(Y, mapped(X))) \tag{35}$$

The final instruction is that the maps are similar but not identical. In the version of the task which we are modelling, we take this to mean that any landmark may be missing from either map. This is coded by creating a mutual default assumption that the two maps are the same.

$$\forall Y : agent\ W : landmark\ mk(def(10, bel(Y, mapped(W)))) \tag{36}$$

The `bel` operator could be left out of this axiom, as after compilation there will be defaults at each level of nesting that any landmark the system knows about is mapped, but since `mapped` is a virtual predicate it is meaningless at the domain level (which is included in the semantics of `mk`). Default strength 10 is a random choice that has no effect since this is the only type of default in the system.

5.3 Compilation

The model is compiled according to the step-by-step process outlined in subsection 3.2. It should be fairly straightforward to see what happens to each of the rules listed above; one step that may not be obvious is the way in which the mutual knowledge that the route giver initially has all the route sections and the route follower none of them is translated into the correct set of individual nested beliefs at each level (subsection 3.2.4). While the `mk` operators are expanded into nested sets of beliefs amongst agents in general, the innermost belief operators refer to the specific types of agent `route_giver` and `route_follower`. These will only unify with the agent lists corresponding to the belief levels represented in the rapid model if the innermost agent is of the right sort, i.e., whatever sort the system is for `obj` and `rmb` beliefs, and whatever sort the user is for `bel` beliefs. Thus if the model is compiled with the system as Doris (a route giver) and the user as Fred (a route follower) the following clauses are generated in respect of initial beliefs about the route:

```
[[obj + [mapped [route_section 30]]]]
[[bel - [mapped [route_section 31]]]]
[[rmb + [mapped [route_section 32]]]]
```

(The integers are added to identify the individuals as separate implicitly universally-quantified variables).

At this stage the ATMS is empty, as no expressions have yet been queried.

5.4 Operation of the compiled model

The operation of the compiled model basically consists of the actions needed to keep the contents of the ATMS in accordance with their specification in subsection 3.3.1, as illustrated in figure 2. Reasoning within the model is thus triggered by updates (resulting from utterances) and queries (generated by the system's planning and plan recognition processes).

5.4.1 Processing of queries

We will deal with the way queries are processed first, because all the model's operation relates to the ATMS contents and nothing is put into the ATMS until queries are made. Let us suppose in our example that when it is in its initial state we make a query to find out whether the system (here modelling Doris, the route giver) believes the user (Fred, the route follower) believes the System has the swamp vivid. The answer we are expecting is yes at default strength 10 (the strength of the default that landmarks appear on both maps), since Doris has the swamp and will assume Fred has it too and will himself assume she has it.

The query is sent to the belief model as a FEBL expression:

$$bel(doris, bel(fred, bel(doris, vivid(swamp\_0)))) \tag{37}$$

This is passed through the compiler and translated into the rapid language as described in subsection 3.2; the result is a query expression at the RMB level.[1]

```
[rmb + [vivid swamp_0]]
```

The next stage involves extending (or, as in this case, creating) the network representation of the modelled agent's reasoning which is used to determine what is believed and to what strength. The term corresponding to the query is not found to be present in the network at the start, so it is used to initiate a backward-chaining search of the rule base. Two rules are found with this non-negated term; one specifying that the term follows from RMBs that there is a description of the swamp's location which is itself vivid, and the other that it follows from the RMB that the swamp is mapped.

The criterion for continuing a backwards-chaining search is that the terms driving the search contain no implicitly quantified variables. The antecedents of the first rule fail this condition and are abandoned, though if a node matching either of them had been present in the network a new rule would have been

[1]Although the assertion of an equivalent term on its own could not generate an RMB, the term is subsumed by the RMB so this is what is queried.

generated by unification, and the other antecedent would be variable-free and used for further searching. The antecedent of the second rule is already variable-free and thus drives further search.

This uncovers default rules which specify that the RMB that the swamp is mapped follows from a singly-nested belief that it is mapped, and that this follows from Doris' own belief that it is mapped, which is itself a premise. If, as in this case, backwards chaining leads to a set of premises that between them entail the queried node given a certain set of defaults, then the premise nodes are created in the network and a forward search is carried out to generate all the consequences of these terms (this will include the queried node itself and any nodes following from it). The results of this forward search do not lead to any contradictions in this instance, so they are not added to the ATMS; however, as was the case with the backward search, new clauses resulting from unifying the search terms with existing clauses are added to the rule base. For instance, the rule base initially contains this clause, relating single-place descriptions to vividness of their subjects at the `rmb` level:

```
[[rmb - [vivid [landmark 133]]] [rmb + [vivid [to_1 relation_1
          [landmark 133]]]]]
```

The query described here will result in the addition of a new clause in which `[swamp_0]` is substituted for `[landmark 133]`.

5.5 Processing of updates

When a new proposition is added to the model by the host system, the reasoning that must be done is merely that required to make sure that all nodes in the network continue to be present and connected as specified in subsection 3.3.1. Suppose the route giver modelled by the system says "The first part of the route goes left of the swamp"; if the term `[bel + [vivid section_1]]` has previously been queried, the forward search initiated by this utterance will result in a new justification for it; if not, it will simply result in new rules being added that will be discovered and linked into such a justification if this term is subsequently queried. However if the partner subsequently replies "I do not understand", the forward-chaining step turns up a contradiction with a term expressing the partner's vividness about the first part of the route. This will initiate backward chaining from this term no matter what has so far been queried, which will turn up a justification for it from the previous utterance.

The result of the process is the representation in the ATMS of all chains of justification from premises or assumptions to queried terms or contradictions. Figure 5 shows a schematic of the network state that would result after the two utterances mentioned above had occurred, if at some point the term `[bel + [vivid swamp_0]]` has been queried. (This term would be believed as a default if queried initially, but after the utterances it cannot be believed as its assumption leads to a contradiction).

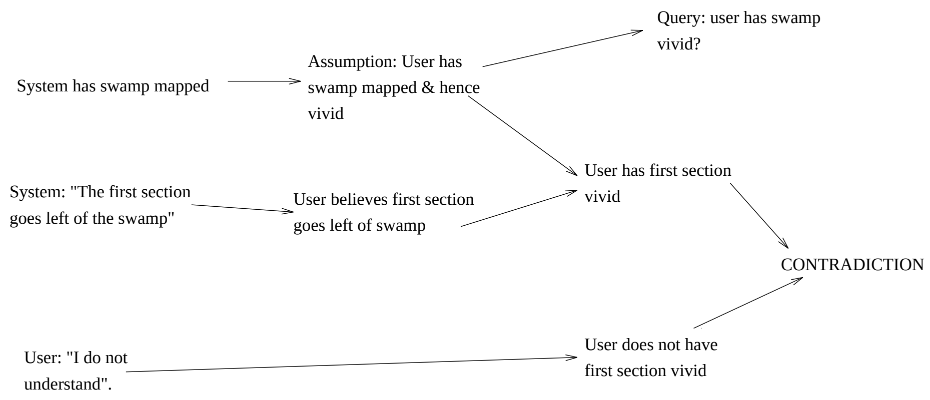

Figure 5: Connections between premises, contradictions and queries

5.6 Change over time

As a result of the route follower's first utterance in this example, the node for `[bel + [vivid section_1]]` has become contradicted directly by a node with the empty environment in its label, i.e., one which is believed regardless of any assumptions. Thus this term cannot be believed no matter what has gone before; if it previously was believed subject to an assumption, this assumption is made a *nogood* in the ATMS and removed.

However, if something subsequently happens which should result in the term being believed again, this should be allowed. Suppose the route giver's next utterance is: "The swamp is below the palm beach". Since the route giver can still assume the route follower has the palm beach, she should conclude that he can thus get the swamp vivid via the description, and hence the first section as well. This happens because of the implementation of the temporal logic of beliefs described in subsection 4.3. It can be seen from the sort declarations above that the sort of `[bel + [vivid section_1]]` is `monotonic`. When a new proposition is added, terms in the ATMS that are not `fixed` are given new data (identity attributes), and new nodes created with the original data. If these are `monotonic` they are justified directly from their previous incarnations; vice versa if they are `contingent`. Other justifications between old and new incarnations include weak persistence assumptions in their antecedents.

So after the route giver's second utterance, the nodes corresponding to the route follower having the swamp and the first section vivid are re-created. The addition of the second utterance causes new justifications for them to be generated. Of course the contradiction with the route follower's previous utterance is also regenerated, but since the term corresponding to its occurrence is of type `transient`, its current incarnation has a weak persistence assumption included in its justification. If the status of `[bel + [vivid section_1]]` is queried again at this point, the evaluation process will find this assumption in contradiction with the route giver's assumption that the route follower has the swamp mapped, and the latter will take precedence.

6 Conclusion and Future Work

In this paper we have described the purpose, design, implementation and functionality of a compiler-based domain-independent belief modelling system. We have illustrated the possibility of applications of such a general-purpose belief modelling system in applications such as student modelling in ITS, simulated agent modelling and general user modelling in HCI.

The process of optimizing the behaviour of the belief model for computational efficiency by exploiting the limitations inherent in humanlike reasoning has been shown to improve the performance of the model over that of an implementation in full of a formal logic. However, it would be possible to improve further on the currently implemented system by optimizing all aspects of the belief-modelling process, most importantly unification between clauses held in the rule base and ATMS components.

The implementation of circumscription or CWA-based reasoning is an important issue in belief modelling, because humans appear to apply such reasoning to everyday problems. To date, computational models of such processes have been unwieldy. While some exploration of the possibility of including such reasoning has been made in the implementation of our compiler, this is another area requiring further work.

The compiler and examples, together with the examples of the restricted natural language input for the Map Task model, have been written in POP-11 (Barrett, Ramsay, and Sloman 1987) and run under Poplog making use of an ATMS implementation in the same language. They can be downloaded from `http://www.cogsci.ed.ac.uk/hcrc/software/BELMOD-compiler/`

A Definitions of components of domain and model specifications

A.1 Sort tree

In the following specification square brackets in the sort tree components have been replaced by curly brackets, as the former are part of the BNF language.

```
tree  ::=  {<individual> [{subtree-list}]}
subtree-list  ::=  subtree | subtree subtree-list
subtree ::= {sort [{subtree-list}]}
sort ::= (unique id word)
```

A.2 sort declarations

```
decl-list ::= decl | decl decl-list
decl ::= sort func-pred [sort-list]
```

```
sort-list ::= sort | sort sort-list
func-pred ::= (unique id word)
```

A.3 domain formulae

```
domain-spec ::= {formula} | {formula & domain-spec}
formula ::= {[quantifier-list] expression}
quantifier-list ::= quantifier | quantifier quantifier-list
quantifier ::= scope variable : sort
scope ::= <all> | <exists>
variable ::= (unique id word)

expression ::= variable | func-pred | {func-pred arg-list} |
-{expression} | {expression infix expression}
arg-list ::= expression | expression arg-list
infix ::= <&> | <=>> | <v>
```

B Grammar for restricted English specification of Map Task axioms

B.1 CFG derived from sort declarations

```
/* Here's a first attempt at a grammar for the Map Task. Naturally its
format is slightly more NL-like than what would be generated by a
grammar derived exclusively from the logical properties of the task
representation. */

[ ;;; sentence and part-of-speech formats
[s
    [pass_modal_vp real_prop]
    [def_op space_int real_prop]

    [neg real_prop]
    [if_syn condition then_syn s]

    [agent_p act_vp]
    [map_item_p pass_vp]

    [det s_word holds_syn]
    ]

[prop
    [real_prop]
    [virt_prop]
    ]
```

```
[real_prop
    [that_syn s]
    ]

[virt_prop
    [where_syn location_p is_syn]
    [how_syn map_item_p appears_syn]
    [whether_syn s]
    ]

[location_p
    [map_item_p]
    [pp]
    ]

[condition
    [s]
    [condition and_syn s]
    ]

[pass_modal_vp
    [pass_modal_op]
    ]

[act_vp
    [doing_vp]
    [modal_vp]
    ]

[pass_vp
    [is_syn pp]
    ]

[modal_vp
    [modal_op prop]
    ]

[doing_vp
    [utter_inform real_prop]
    [utter_report virt_prop]
    [makes_syn utter_move]
    ]

[utter_move
    [det utter_word]
```

```
    ]

[pp
    [det pp_word]
    [prep1_p map_item_p]
    [prep2_p map_item_p conj map_item_p]
    ]

[prep1_p
    [prep1]
    [det prep1_word]
    ]

[prep2_p
    [prep2]
    [det prep2_word]
    ]

[agent_p
    [agent_name]
    [det agent_word]
    [personal_pronoun]
    ]

[map_item_p
    [landmark_p]
    [route_section_p]
    [det map_item_word]
    ]

[landmark_p
    [landmark_name]
    [det landmark_word]
    ]

[route_section_p
    [route_section_name]
    [det route_section_word]
    ]

] -> grammar;

[ ;;; lexical items (some multiword)
[modal_op knows]
[pass_modal_op it_is_mutual]
[def_op default_in_space]
```

```
[if_syn if]
[is_syn is]
[how_syn how]
[and_syn and]
[that_syn that]
[then_syn then]
[makes_syn makes]
[holds_syn holds]
[where_syn where]
[appears_syn appears]
[whether_syn whether]

[s_word fact other_fact]
[pp_word place]
[prep1_word point_near]
[prep2_word point_between]

[det the a an some]
[neg it_is_false]

[utter_word utterance]
[utter_inform asserts acknowledges concludes]
[utter_report queries misunderstands affirms denies]

[agent_name fred doris]
[personal_pronoun he she]
[agent_word agent other_agent]
[prep1 above left_of below]
[prep2 between astride]
[route_section_name section_1 section_2]
[landmark_name the_palm_beach the_swamp]

[map_item_word map_item other_map_item]
[route_section_word route_section]
[landmark_word landmark]
[space_int 0 1 2 3 4 5 6 7 8 9 10]
] -> lexicon;
```

B.2 Parse tree from sample utterances

"It is mutual that if an agent queries where a map item is then it is false that he knows where the map item is."

```
[[s [pass_modal_vp [pass_modal_op it_is_mutual]]
    [real_prop [that_syn that]
```

```
      [s [if_syn if]
        [condition [s [agent_p [det an]
                               [agent_word agent]]
               [act_vp [doing_vp [utter_report queries]
                        [virt_prop [where_syn where]
                             [location_p [map_item_p [det a]
                              [map_item_word map_item]]
                             [is_syn is]]]]]]
        [then_syn then]
        [s [neg it_is_false]
         [real_prop [that_syn that]
               [s [agent_p [personal_pronoun he]]
                [act_vp [modal_vp [modal_op knows]
                         [prop [virt_prop [where_syn where]
                                [location_p [map_item_p [det the]
                                   [map_item_word map_item]]]
                                [is_syn is]]]]]]]]]]]
```

"It is mutual that if an agent concludes that he knows where a map item is then he knows where the map item is." (Here, "concludes" is one of the utterance types, one that might be instantiated by "Okay, I've got that now".

```
[[s [pass_modal_vp [pass_modal_op it_is_mutual]]
    [real_prop [that_syn that]
      [s [if_syn if]
        [condition [s [agent_p [det an]
                               [agent_word agent]]
               [act_vp [doing_vp [utter_inform concludes]
                        [real_prop [that_syn that]
                             [s [agent_p [personal_pronoun he]]
                               [act_vp [modal_vp [modal_op knows]
             [prop [virt_prop [where_syn where]
                     [location_p [map_item_p [det a]
                      [map_item_word map_item]]]
                     [is_syn is]]]]]]]]]]]
        [then_syn then]
        [s [agent_p [personal_pronoun he]]
         [act_vp [modal_vp [modal_op knows]
                  [prop [virt_prop [where_syn where]
                          [location_p [map_item_p [det the]
                           [map_item_word map_item]]]
                          [is_syn is]]]]]]]]
        ]
```

References

Allen, J. F. and G. Ferguson (1994). Actions and events in interval temporal logic. Technical Report TR-521, University of Rochester, Department of Computer Science, Rochester, New York.

Anderson, A. H., M. Bader, E. G. Bard, E. Boyle, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. Thompson, and R. Weinert (1991). The HCRC Map Task Corpus. *Language and Speech 34*(4), 351–366.

Appelt, D. E. (1985). *Planning English Sentences.* Cambridge University Press.

Barrett, R., A. Ramsay, and A. Sloman (1987). *POP-11: A practical language for artificial intelligence.* Chichester: Ellis Horwood.

Brachman, R. J. and J. G. Schmolze (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science 9*(2), 171–216.

Byrne, R. M. J. (1991). The model theory of deduction. In Y. Rogers, A. Rutherford, and P. A. Bibby (Eds.), *Models in the Mind.* Academic Press.

Carletta, J. C. (1992). *Risk-Taking and Recovery in Task-Oriented Dialogue.* Ph. D. thesis, Edinburgh University Department of Artificial Intelligence.

Carpenter, P. A., A. Miyake, and M. A. Just (1994). Working memory constraints in comprehension: Evidence from individual differences, aphasia, and aging. In M. Gernsbacher (Ed.), *Handbook of Psycholinguisitics.* San Diego, CA: Academic Press.

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence 32*(3), 333–377.

Cohen, P. R. and H. J. Levesque (1987, February). Persistence, intention and commitment. Technical Report 415, Artificial Intelligence Center, SRI International. Reprinted as (Cohen and Levesque 1990a).

Cohen, P. R. and H. J. Levesque (1988, April). Rational interaction as the basis for communication. Technical Report 433, Artificial Intelligence Center, SRI International. Reprinted as (Cohen and Levesque 1990b).

Cohen, P. R. and H. J. Levesque (1990a). Persistence, intention and commitment. In P. R. Cohen, J. Morgan, and M. E. Pollack (Eds.), *Intentions in Communication*, pp. 33–69. Cambridge, MA: MIT Press.

Cohen, P. R. and H. J. Levesque (1990b). Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack (Eds.), *Intentions in Communication*, pp. 221–255. Cambridge, MA: MIT Press.

de Kleer, J. (1986). An assumption-based truth maintenance system. *Artificial Intelligence 28*(2), 127–162.

Etherington, D. W., A. Borgida, R. J. Brachman, and H. Kautz (1989). Vivid knowledge and tractable reasoning: Preliminary report. In *Proceedings of IJCAI-89*, pp. 1146–1152.

Frisch, A. M. (1987). Inference without chaining. In *Proceedings of IJCAI-87*, pp. 515–519.

Frisch, A. M. (1991). The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence 49*, 161–198.

Geissler, C. and K. Konolige (1986). A resolution method for quantified modal logics of knowledge and belief. In J. Y. Halpern (Ed.), *Conference on Theoretical Aspects of Reasoning About Knowledge*, pp. 309–324.

Ginsberg, M. L. (1988). Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence 4*, 265–316.

Grice, H. P. (1975). Logic and conversation. In P. Cole and J. L. Morgan (Eds.), *Syntax and Semantics 3: Speech Acts*, pp. 41–58. New York: Academic Press.

Halpern, J. Y. and Y. Moses (1985). A guide to the modal logics of knowledge and belief: Preliminary draft. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Volume 1, pp. 480–490.

Hewson, C. (1994). Empirical evidence regarding the folk psychological concept of belief. In A. Ram and K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Hillsdale, New Jersey, pp. 403–408. Lawrence Erlbaum Associates.

Jiang, Y. J. (1994). Multiagent autoepistemic logic — an extrospective view. In J. Doyle, E. Sandewall, and P. Torasso (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference*. Morgan Kaufmann.

Kaplan, D. (1969). Quantifying in. In D. Davidson and J. Hintikka (Eds.), *Words and Objections*. Dordrecht, NL: Reidel.

Kelleher and van der Gaag (1993). The LazyRMS - avoiding work in the ATMS. *Computational Intelligence 9*(3), 239–253.

Kobsa, A. and W. Pohl (1995). The user modeling shell system BGP-MS. *User Modeling and User-Adapted Interaction 4*(2), 59–106.

Konolige, K. (1986). *A Deduction Model of Belief*. Los Altos, CA: Morgan Kaufman.

Konolige, K. (1988, August). Hierarchic autoepistemic theories for nonmonotonic reasoning: Preliminary report. Technical Report 446, Artificial Intelligence Center, SRI International.

Lakemeyer, G. (1987). Tractable meta-reasoning in propositional logics of belief. In *Proceedings of IJCAI-87*, pp. 402–408.

Levesque, H. J. (1984). A logic of implicit and explicit belief. In *Proceedings of AAAI-84*.

McAllester, D. (1980). An outlook on truth maintenance. Technical Report AIM-551, MIT Artificial Intelligence Laboratory, Cambridge, MA.

McDermott, D. V. (1982). A temporal logic for reasoning about processes and plans. *Cognitive Science 6*, 101–155.

Moore, R. C. (1985). A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore (Eds.), *Formal Theories of the Commonsense World*, pp. 319–358. Norwood, New Jersey: Ablex.

Moore, R. C. (1988). Autoepistemic logic. In P. Smets, E. H. Mandani, D. Dubois, and H. Prade (Eds.), *Non-Standard Logics for Automated Reasoning*. Academic Press.

Ohlbach, H. J. (1991). Semantics-based translation methods for modal logics. *Journal of Logic and Computation 1*(5), 691–746.

Pereira, F. C. N. and R. N. Wright (1991). Finite-state approximation of phrase-structure grammars. In *29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, California, pp. 246–255.

Reiter, R. and J. de Kleer (1987). Foundations of assumption-based truth maintenance systems: preliminary report. In *Proceedings of the American Association for Artificial Intelligence*, Volume 1, pp. 183–187. Morgan Kaufmann.

Selman, B. and H. Kautz (1991). Knowledge compilation using horn approximations. In *Proceedings of the AAAI*, pp. 904–909.

Shoham, Y. (1988). *Reasoning About Change*. Cambridge, MA: MIT press.

Stenning, K. (1991). Distinguishing conceptual and empirical issues about mental models. In Y. Rogers, A. Rutherford, and P. A. Bibby (Eds.), *Models in the Mind*. Academic Press.

Taylor, J. (1994a). *A Multi-Agent Planner for Modelling Dialogue*. Ph. D. thesis, School of Cognitive and Computing Sciences, University of Sussex.

Taylor, J. (1994b, November). Using hierarchical autoepistemic logic to model beliefs in dialogue. Technical Report HCRC/RP-60, Human Communication Research Centre, University of Edinburgh.

Taylor, J., J. Carletta, and C. Mellish (1996). Requirements for belief models in cooperative dialogue. *User Modelling and User-Adapted Interaction 6*.

Walther, C. (1987). *A Many-Sorted Calculus Based on Resolution and Paramodulation*. Los Altos, CA: Morgan Kaufman.