A Computational Model of Belief

by

Aaron Nathan Kaplan

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Lenhart K. Schubert

Department of Computer Science
The College
Arts and Sciences

University of Rochester
Rochester, New York

2000

Curriculum Vitae

Aaron Kaplan was born in Beckley, West Virginia on September 8, 1970, and grew up in the Syracuse, New York area. He attended Cornell University from 1988 to 1992, and received the degree of Bachelor of Arts with distinction. In 1992 he worked at Eloquent Technology in Ithaca, New York on the development of speech synthesis software, and in 1993 he worked as a technical writer at ILOG, a software company in Paris, France. He entered the University of Rochester in 1993, and began working in natural language processing and knowledge representation under the supervision of Professor Lenhart Schubert. He received the degree of Master of Science in Computer Science in 1995.

Acknowledgments

The most important thing about my stay in Rochester has been that for several years, my time was my own, and so I was forced to decide what I actually wanted to do. I know I'll look back with fondness on the days when I had that freedom, but often it made me miserable. If I had been working a nine-to-five job, I might have blamed my unhappiness on external forces, but here I ended by understanding what it takes to feel fulfilled.

I gather that not everyone's grad school experience is like this. With a different advisor, I might have graduated much sooner, but come out the same person who went in. My greatest regret is that I didn't take advantage of Len's insight and excitement as often as I did his patience. The same goes for my other committee members, James Allen and David Braun.

The growth I underwent here was by no means purely academic. I learned more from my friends, most notably Gabriela Galescu and Karen LaMacchia, than from anyone else.

Finally, there is my family: Mom, Dad, and Becky. Unlike anyone or anything else, they were never in doubt.

This work was supported by NSF research grants numbers IRI-9623665 and IRI-9503312, and U.S. Air Force/Rome Labs research grant number F30602-97-1-0348.

Abstract

We propose a logic of belief in which the expansion of beliefs beyond what has been explicitly learned is modeled as a finite computational process. The logic does not impose a particular computational mechanism; rather, the mechanism is a parameter of the logic, and we show that as long as the mechanism meets a particular set of constraints, the resulting logic has certain desirable properties. Chief among these is the property that one can reason soundly about another agent's beliefs by simulating its computational mechanism with one's own.

The EPILOG system, a computer program designed for narrative understanding, serves as a case study for the application of the model and the implementation of simulative inference about belief.

Table of Contents

List of Figures

1 Introduction

The human mind is a complex system about whose internal workings we understand very little. Nevertheless, we can predict with a useful degree of accuracy how a normal person will behave in quite a wide range of circumstances. Without this ability for prediction, much of our everyday social interaction would be impossible. There are two ways we might make this type of inference: by analogy to people we have observed in similar situations before, or by analogy to ourselves, using a kind of introspection: "If I were in such a situation, I would ...." This dissertation concerns the latter mode of inference, which we call *simulative inference*.

In particular, we focus on simulative inference about belief, *i.e.* inference that follows the following pattern: "α believes $\varphi_1, \ldots, \varphi_n$; if I believed those things, then I would also believe ψ; therefore, α believes ψ."

Our approach is to define a logic of belief. That is, we propose a system consisting of the following things:

1. a language in which facts about what various agents believe can be expressed,
2. a proof theory, *i.e.* a set of rules (including a rule of simulative inference) by which, given a set of premises expressed in the language, we can find conclusions that follow from them,
3. a model theory, *i.e.* a systematic definition of the conditions under which any sentence of the language is true.

Ours is by no means the first formal model of belief to be proposed. There is a substantial body of work in logic and the philosophy of language that is directly related, and there have been contributions from our own field of artificial intelligence, including some concerned with the issue of simulative inference. However, the theory of belief has not yet fully assimilated the central idea of artificial intelligence, namely the idea that intelligence, and therefore belief, is a property demonstrated by computational systems. We aim to demonstrate that by incorporating this point of view, the

formal theory of belief can be brought closer to matching intuitions and empirical observations. Furthermore, our model provides a framework in which to investigate the technique of simulative inference. This framework allows us to show that simulative inference can sometimes yield incorrect results, and (more importantly) to give a precise characterization of circumstances under which it is guaranteed to yield correct results.

1.1 Perspective and Motivation

Our work is in the paradigm of logical artificial intelligence, a tradition that stems from McCarthy's 1958 "Advice Taker" paper [1958]. The basic aim of logical AI is to build systems that store and manipulate expressions of a logic in a way that emulates human thought. For example, a system might represent the facts that John is at his desk, that the desk is at his home, and that "at" is a transitive relation, by the following:

$$at(john, desk)$$
$$at(desk, home)$$
$$at(x, y) \wedge at(y, z) \supset at(x, z)$$

and the system could be equipped with an inference mechanism by which it would derive the new sentence

$$at(john, home),$$

meaning that John is at home, from the previous sentences. The central hypothesis of logical AI is that if we are eventually able to encode enough of human common sense knowledge in logical form, and we develop sufficiently sophisticated inference mechanisms, then this mechanical manipulation of symbols will become functionally indistinguishable from intelligence (whether it would actually *be* intelligence is a controversial question, on which we will not venture an opinion here).

Of course, the idea of describing or implementing reasoning as manipulation of symbols predates the field of computer science. Symbolic logic was developed for precisely this purpose—making inference a mechanical operation that proceeds according to fixed rules. McCarthy's ideas, sparked by the advent of machines capable of carrying out logical derivations automatically, were not so much an innovation as a change of focus. First of all, logical AI uses symbolic logic for everyday, common sense kinds of reasoning, whereas it had typically been used for analyzing more formal types of arguments, such as those found in mathematics. And second, being a branch of computer science, AI puts special emphasis on the process of performing inference, *i.e.* on effective methods for finding a proof of a given conjecture, or for identifying the most interesting or useful conclusions that can be drawn from a set of premises, without being sidetracked by the infinitely many less interesting ones.

The logical approach is not the only approach to AI, and some advocates of other approaches have been critical of this style of research. In particular, there is a trend

towards attacking the AI problem with a more bottom-up strategy, first understanding low-level behaviors such as simple perception and motor control, then eventually combining models of those various behaviors to build models of higher-level ones, and so on in a hierarchical fashion until a model of human-level intelligence is reached. This is a reasonable methodology; it seems indisputable that for an agent to display intelligent behavior, it must have facilities for interacting with its world. Furthermore, there has been encouraging success in implementing some of the lowest-level behaviors. However, contrary to the claims of some of its practitioners, the bottom-up approach has in no way supplanted or discredited symbolic AI. The successes that the bottom-up approach has had so far are at tasks very different from those addressed by the symbolic approach, and the path from there to a comprehensive model of human intelligence is far from clear. In one sense, the bottom-up approach could be seen as more general: given the assumption of materialism that is implicit in all of AI, it must be true that in theory a comprehensive model of a human mind could be built up of neurologically-based models of brain functions. But we do not currently have anything near that level of understanding of the brain, nor is there reason to think that such an understanding is forthcoming in our lifetime, if ever, so for the time being, curiosity about high-level behaviors like planning and language use can only be addressed by high-level research divorced from the reality of neural implementation. Symbolic AI is to the AI of perception and motor control as sociology and economics are to psychology. No truly complete theory of social behavior can exist without a complete theory of individual behavior to build on, but in the absence of a complete theory of the individual, sociologists and economists continue to do what they can, because the social questions are the ones that they find most compelling. Likewise, theories of language use and other high-level human behavior might never be truly complete until we understand at a low level how the brain works, but in the meantime, we do the best we can with the tools we have. We know that our efforts alone won't lead to a grand unified theory of intelligence, but they further our understanding of intelligence nonetheless, and sometimes that understanding can even be put to practical use.

1.2 Belief

Logical AI requires the precise codification of concepts usually understood only loosely, so that they can be used as premises and rules of inference in a symbolic logic. The concept of belief is one that has been considered at length by philosophers, so there is a substantial body of existing work to inform our design. Chapter 2 includes a summary of that work, with bibliographic references; we describe here only enough to explain what we consider to be the main shortcomings of existing models, and to sketch our proposed solution.

In the most straightforward models of belief, belief is simply a relationship that can hold between a believer and either a sentence or a proposition. The distinction

between sentences and propositions will be explained in Section 2.5.1, but need not concern us at the moment. These models, in their simplicity, license very little in the way of inference about belief. In these models, a person can believe that roses are red and violets are blue, without believing that roses are red. "Roses are red and violets are blue" is one sentence (denoting one proposition), and "roses are red" is a different sentence (denoting a different proposition), and a person can perfectly well stand in the belief relationship to one of them and not the other. This is not to say that these models are incompatible with a theory of how belief in one sentence (or proposition) is related to belief in another, only that they do not themselves include such a theory.

Another model, the "possible worlds" model, has more structure. In this model, the set of sentences that a person believes can't be any arbitrary set—it must be one that is closed under logical consequence. That is, if φ is a consequence of someone's beliefs, then φ is itself one of his beliefs. Since "roses are red" follows from "roses are red and violets are blue," anyone who believes the latter must also believe the former. This may seem attractive, particularly as the substrate for a theory of simulative reasoning: if John believes φ, and believing φ would cause me to believe ψ as well, then according to the possible worlds model I am justified in concluding that John believes ψ (assuming that my leap from φ to ψ is justified). However, the possible worlds model is unreasonable from a computational standpoint. It is well known that for any sufficiently expressive logic, the question of entailment is undecidable. This means that there is no algorithm which, given an arbitrary set of premises and a conjecture, is guaranteed to tell us after a finite amount of time whether the conjecture follows from the premises. In other words, in the possible worlds model people are perfect reasoners, "logically omniscient" in a way that real people provably can't be (assuming we accept that what goes on in a brain is computation).

The right model would seem to be between these two extremes. If a real person believes φ, then he also believes any easily discovered consequences of φ, but if there are some obscure and difficult to discover consequences, he might not believe those. The question is then how to define which consequences are easily discovered and which are not. There have been a number of proposals along these lines, many of them from the AI community. The details vary considerably, but many of the proposals eliminate logical omniscience by somehow eliminating the rule of *modus ponens* from the implicit reasoning capacity of believers. These models achieve the desired result to some degree. Now anyone who believes that roses are red and violets are blue also believes that roses are red, but unlike in the possible worlds model, a person can know[1] the rules of chess yet not know whether there is a strategy that guarantees white a win. But something is still wrong. Because *modus ponens* is out, a person can believe that Fido

[1]In this work, we use the words "know" and "believe" interchangeably. The precise definition of knowledge is somewhat controversial, but it is generally accepted that knowing φ entails at least believing φ. Our work concerns only the more primitive concept of belief, but we occasionally use the word "know" in contexts where "believe" seems unnatural or might be read as having connotations that are not intended.

is a doberman, and believe that dobermans are dogs, but not believe that Fido is a dog. This seems unreasonable.

It is our assertion that these models fail to satisfy intuitions about what it really means to believe something because they attempt to define "easy" inferences in terms of properties intrinsic to the inferences themselves. In fact, we argue, an inference is not inherently easy or difficult. Rather, it is easy or difficult *for someone*, for a person who is looking for conclusions to draw. The mind is attuned to make certain kinds of inferences quickly and automatically, and to ignore other possibilities. Therefore, an accurate model of belief must include a model of a mind. Partial as we are to the concepts of AI, we take this to mean a model of a computational mechanism.

In our model of belief, a believer has a *belief machine*, which is an abstraction of a computational mechanism for information storage and retrieval. The machine's behavior is described by two recursive functions, *ASK* and *TELL*. Each is a function of two arguments, the first being a state of the machine, and the second a sentence of a logic. The value of *ASK*(S, φ) is either yes or no, indicating whether an agent whose belief machine is in state S believes the sentence φ or not. *TELL* is the machine's state transition function: upon receiving a new fact φ, a machine in state S will move to state *TELL*(S, φ). It may or may not believe φ in this new state, *i.e.* it may or may not accept the proffered fact. Earlier, we characterized simulative inference as following this pattern: "α believes $\varphi_1, \ldots, \varphi_n$; if I believed those things, then I would also believe ψ; therefore, α believes ψ." In the *ASK* and *TELL* framework, the condition "if I believed $\varphi_1, \ldots, \varphi_n$ then I would also believe ψ" is verified by simulation: we *TELL* the belief machine φ, and then *ASK* it about ψ, and it answers yes.

Obviously, we do not intend to give in this dissertation a thorough functional description of the mechanisms humans use to maintain their beliefs. Rather, we define a class of logics such that, given an arbitrary computational mechanism for belief storage and retrieval, there is a logic for reasoning about the beliefs of agents who use that mechanism. The human mind implements some complex algorithm for maintaining beliefs, and science has as yet been unable to identify that algorithm; but whatever it might be, if it can be described in the *ASK* and *TELL* framework, then it defines a logic of belief.

The proposal that a model of belief should include a model of inferential ability is not completely new. The idea has appeared in the literature in various forms from time to time. What is novel about this work is the treatment of simulative inference within our computational model of belief. To our knowledge, there has been only one other proposal that includes a formalization of simulative inference with a rigorous semantic justification, namely Konolige's deduction model of belief. The essential difference between his theory and ours is the model of inferential computation that is used: we allow an arbitrary algorithm, while Konolige requires the exhaustive application of a set of deductive inference rules. In Section 5.5 we make a detailed comparison of our model with the deduction model.

1.3 Simulative Inference

In order to study simulative inference, we make the assumption that all agents' belief machines are functionally identical, *i.e.* that if two agents have different beliefs, it is only because they have learned different things, not because they have different inherent abilities. This assumption would be accurate for identically constructed artificial agents, but also seems to us a reasonable first approximation about human reasoners. The requirement of functional identity is not as strong as it might first appear—the fact that two agents have the same belief machine does not necessarily mean that they use the same inference methods. What goes on in the belief machine in response to a series of input formulas could be any sort of computation, including the learning of new inference rules. Functional identity is also a particularly benign requirement when the beliefs being studied are those that arise in the course of communication via language. Often, part of the information that people wish to convey goes unsaid, because the speaker can rely on the hearer making certain inferences. An assumption of similar inferential ability is precisely what is required for this type of communication.

Our original interest in simulative inference stemmed from work on EPILOG, a computer system for knowledge representation and reasoning in support of language understanding. We have given the system the ability to reason about the beliefs of the people in narratives it has been given. One might be particularly concerned about the assumption of functional identity in this case, because given the limitations of our current understanding of human intelligence, EPILOG's inferential ability is certain to be quite different from that of a human. However, whatever its failings, EPILOG, like any other AI system, is intended to be an approximation of human mental functioning. The simulative inference it performs about the beliefs of humans will be correct to whatever extent the approximation is successful; and as science progresses and the approximation is improved, the accuracy of simulative inference will improve commensurately.

Even given the assumption that one has access to a belief machine functionally identical to that of the believer about which one is reasoning, simulative inference is not always guaranteed to give correct results. We will need to introduce the details of our model before we can make this assertion more concretely, but an example will illustrate the sort of problem that can arise. A belief machine's *ASK* computation must be guaranteed to halt eventually on any input. Consider a belief machine that satisfies this condition by placing a time bound of five seconds on *ASK* computations. If it can confirm within five seconds that a query sentence follows from what it knows, then it answers yes, otherwise it answers no. Simulative inference with this machine can yield conclusions that do not necessarily follow from the premises. For example, say that both ψ and χ follow from φ, and that having been *TELL*ed φ alone, the belief machine can see in less than five seconds that ψ follows, but can't see in five seconds that χ follows. Assume further that if the machine is *TELL*ed both φ and ψ, then with the work of inferring ψ already done, it can make the remaining leap to χ in less than five seconds. Let us say that we know at first only that someone believes φ. We can

reason by simulation as follows: "He believes φ; if I believed φ, I would also believe ψ; therefore, he believes ψ". Now, given the information that the person believes ψ, we could do another simulative step as follows: "He believes φ and ψ; if I believed φ and ψ, I would also believe χ; therefore, he believes χ." From the single premise that the person believes φ, we have concluded in two steps that he also believes χ. But this is not a valid inference. It is possible, given our original premise, that the person's belief machine has been *TELL*ed only φ, and that therefore he believes φ and ψ but not χ. In Section 3.1, we will introduce a set of constraints on the relationship between *ASK* and *TELL*, and demonstrate that if those constraints are satisfied, then in fact simulative inference is sound, *i.e.* guaranteed to yield only conclusions that do follow from the premises. We then address the question of whether those constraints are reasonable, in two respects: whether there are useful inference algorithms that satisfy the constraints, and whether actual human belief can be said to satisfy them.

Artificial intelligence is part science and part engineering. The development of our formal models is motivated by the philosophical goal of gaining a fuller understanding of the world, but also by the practical goal of building working systems. Understanding the conditions under which simulative inference is sound allows us to use the technique in a principled way in a reasoning system. In Chapter 4, we analyze the EPILOG system in the belief machine framework, and examine the consequences of our formal results for the practical matter of adding simulative inference to the system. We also discuss a more pragmatic problem that was an obstacle to making simulative inference in EPILOG efficient, and the solutions we used to overcome it. The problem involves the fact that EPILOG uses various non-sentential knowledge representations (the system's input and output are always in the form of logical sentences, but internally, it uses various non-sentential representations specifically designed for efficient reasoning about particular things). While this part of the work is less theoretical than the formal part, it is of rather general applicability. The problem we identified will affect the implementation of not only simulative inference, but any inference mechanism that depends on keeping track of the system's reasons for believing each stored fact, including truth maintenance systems and probabilistic reasoning systems.

2 The Model

So far, we have only sketched the concepts of our model in intuitive terms. In order to be able to discuss the model and the technique of simulative inference more precisely, in this chapter we give formal definitions of the syntax and semantics of the logic, and formalize simulative inference as an inference rule in the logic. The rule may or may not be sound, depending on the choice of belief machine; we list some natural constraints such that for any belief machine that satisfies them, if belief is defined in terms of that machine, then the simulative inference rule (also using that machine) is sound, *i.e.* from true premises it generates only true conclusions.

In Section 2.4 we add to the logic an indexical term me that can be used to express facts about an agent's beliefs about itself. Among other things, the indexical introduces the possibility of belief machines with introspection, *i.e.* of agents that have knowledge about their own beliefs. We show that given an introspective belief machine, the rule of negative simulative inference can be used in proving positive as well as negative statements about belief, and that therefore a restricted form of completeness can be maintained even without the positive rule. The negative rule, while less natural than the positive one, is sound for a broader class of machines, a class that includes machines that perform default reasoning.

This chapter contains technical material, but only to the extent necessary to transform our intuitions into concrete and precise definitions. We postpone lengthy proofs until the next chapter.

2.1 Syntax, Semantics, and Some Notation

Our model of belief is built around the concept of the *belief machine*, which is an abstraction of a computational inference mechanism. In the model, each agent has a belief machine that it uses for storing and retrieving information. The agent enters facts it has learned into its belief machine, and can then pose queries to it. Input and queries are expressed as logical sentences, but the model does not constrain the form in which the machine stores and manipulates the information internally. For example,

the machine might use diagrammatic or algorithmic encodings of information. The machine may perform some inference in answering queries, but it must be guaranteed to give an answer in a finite amount of time. An agent believes a sentence φ if its belief machine is in a state such that the query φ is answered affirmatively.

A belief machine is characterized by two functions, *TELL* and *ASK*. *TELL* describes how the state of the machine changes when a new sentence is stored: if S is the current state of the belief machine, and φ is a sentence, then the value of *TELL*(S, φ) is the new state the belief machine will enter after φ is asserted to it. The value of *ASK*(S, φ) is either yes or no, indicating the response of a machine in state S to the query φ.

This model of belief is used to interpret sentences of a logic, which consists of ordinary first-order logic (FOL) plus a modal belief operator B. Where α is a term and φ is a formula, $B(\alpha, \varphi)$ is a formula whose intended meaning is that α believes φ. Let the language L be the set of formulas formed in the usual way from the logical constants $\neg$, $=$, $\wedge$, $\vee$, $\supset$, $\forall$, and $\exists$, the modal operator B, and a set of individual constants, predicate constants, function constants, and variables (infinitely many of each). $\perp$ is notation for an arbitrary contradiction $\varphi \wedge \neg\varphi$. L_c is the set of sentences (closed formulas) of L.

Formally, a belief machine is a structure $\langle \Gamma, S_0, \textit{TELL}, \textit{ASK} \rangle$, where

- Γ is a (possibly infinite) set of states,
- $S_0 \in \Gamma$ is the initial state,
- *TELL* : $\Gamma \times L_c \rightarrow \Gamma$ is the state transition function,
- *ASK* : $\Gamma \times L_c \rightarrow \{yes, no\}$ is the query function.

A formula of L has a truth value relative to a model, which is composed of a domain of individuals and an interpretation function, as in a model for ordinary FOL, and a function γ that assigns a belief state to each individual (for simplicity, we do not distinguish between individuals that are believers and ones that aren't). A single belief machine is chosen ahead of time to describe the reasoning abilities of all agents—the belief machine does not vary from model to model. Therefore, concepts such as entailment, soundness, and completeness are only meaningful relative to a particular choice of belief machine. To be explicit about this, we will sometimes refer to an "m-model," where m is a belief machine. Formally, given a belief machine $m = \langle \Gamma, S_0, \textit{TELL}, \textit{ASK} \rangle$, an m-model is a structure $\langle D, I, \gamma \rangle$, where

- D is the domain of individuals,
- I is an interpretation function that maps variables and individual, predicate, and function constants to set-theoretic extensions, as in ordinary FOL,
- $\gamma : D \rightarrow \Gamma$ is a function that assigns each individual a belief state.

We will use the notation $|\tau|^M$ to mean the denotation of term τ under model M. Note that the domain of the interpretation function includes the variables, so that a model assigns denotations to all terms, even those containing free variables. The denotations of functional terms are determined recursively in the usual way.

The truth values of ordinary (non-belief) atomic formulas, and of complex formulas, are determined in the usual way. In particular, a universally quantified formula $\forall\nu\varphi$ is true in a model M if the open formula φ is true in every model M' that differs from M by at most its interpretation of the variable ν; and similarly for existential formulas.

The semantics of "quantifying-in," *i.e.* of a variable that occurs in a belief context but whose binding quantifier is outside that belief context, is handled by way of *variable substitutions*, which are mappings from variables to ground terms (terms containing no variables). A variable substitution σ is *extension-preserving* under model M if, for every variable ν, the denotation of ν and the denotation of $\sigma(\nu)$ are the same in M. We write φ^σ to mean the formula that results from replacing every free variable occurrence in φ with the ground term to which σ maps that variable.

Where $m = \langle \Gamma, S_0, \textit{TELL}, \textit{ASK}\rangle$ is a belief machine, and $M = \langle D, I, \gamma\rangle$ is an m-model, a belief atom $B(\alpha, \varphi)$ is true in M iff there exists some variable substitution σ which is extension-preserving under M such that $\textit{ASK}(\gamma(|\alpha|^M), \varphi^\sigma) = yes$. This gives a variable that occurs in a belief context, but is not bound in that context, a reading of implicit existential quantification over terms with the same denotation as the variable. For example, $B(a, P(x))$ is true in model M if there is some term τ, which denotes the same thing as x in M, for which $B(a, P(\tau))$ is true. This gives a natural interpretation to quantifying-in: $\exists x B(a, P(x))$, which intuitively means "There is something which a believes to be P," is true if there is some individual, and some term τ which denotes that individual, such that a believes the sentence $P(\tau)$.

We will use $\textit{TELL}(S, \varphi_1, \ldots, \varphi_n)$ as an abbreviation for

$$\textit{TELL}(\ldots \textit{TELL}(\textit{TELL}(S, \varphi_1), \varphi_2), \ldots, \varphi_n),$$

i.e. the state that results from successively *TELL*ing each element of the sequence, beginning in state S.

The notation $\mathcal{B} \cdot S$ means the belief set of a machine in state S, *i.e.*

$$\mathcal{B} \cdot S = \{\varphi | \textit{ASK}(S, \varphi) = yes\}.$$

A sentence φ is *acceptable* in state S if *TELL*ing the machine φ while it is in state S causes it to believe φ, *i.e.* if

$$\varphi \in \mathcal{B} \cdot \textit{TELL}(S, \varphi).$$

A sentence φ is *monotonically acceptable* in state S if it is acceptable in S and *TELL*ing the machine φ while it is in state S does not cause it to retract any beliefs, *i.e.* if

$$\mathcal{B} \cdot S \cup \{\varphi\} \subseteq \mathcal{B} \cdot \textit{TELL}(S, \varphi).$$

A sequence of sentences $\varphi_1, \ldots, \varphi_n$ is monotonically acceptable in state S if each element φ_i of the sequence is monotonically acceptable in the state *TELL*$(S, \varphi_1, \ldots, \varphi_{i-1})$. A sequence $\varphi_1, \ldots, \varphi_n$ is acceptable in state S_0 (we do not define acceptability of sequences in other states) if

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \varphi_i) = yes$$

for all $1 \leq i \leq n$, and if all initial subsequences of $\varphi_1, \ldots, \varphi_n$ are also acceptable (defined recursively). That is, a sequence is acceptable if, as each of its elements is *TELL*ed to a belief machine starting in the initial state, the machine accepts the new input and continues to believe all of the previous inputs (though it might cease to believe sentences that were inferred from the previous inputs).

2.2 The Simulative Inference Rule

Simulative reasoning is reasoning of the following form: "α believes $\varphi_1, \ldots, \varphi_n$; if I believed those things, then I would also believe ψ; therefore, α believes ψ." This form of reasoning is expressed by the following inference rule, where the formulas above the line are the premises, the formula below the line is the conclusion, and the rule applies only when the condition written below it holds:

$$\frac{B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)}{B(\alpha, \psi)}$$

if *ASK*$(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$.

While most of our inference rules will apply to all formulas, this rule applies only to sentences, because to apply the rule one must use the sentences in *TELL* and *ASK* computations (recall that *TELL* and *ASK* are defined only for sentences).

This rule may or may not be sound, depending on the choice of belief machine. We will show in Chapter 3 that it *is* sound when the belief machine satisfies the constraints listed below. Though the inference rule explicitly only requires that

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$$

for one ordering of the φ_i, the constraints entail that if the φ_i can all be believed simultaneously, then the order in which they are *TELL*ed is not significant.

C1 (closure) *For any belief state S and sentence φ, if*

$$\textit{ASK}(S, \varphi) = yes$$

then

$$\mathcal{B} \cdot \textit{TELL}(S, \varphi) = \mathcal{B} \cdot S.$$

The closure constraint says that *TELL*ing the machine something it already believed does not increase its belief set (though the belief *state* may change).

This rules out machines such as the one in the example on page 6, the machine that ensures that the *ASK* function always halts by imposing a time limit, answering no when the limit is reached before a definitive answer has been found. Simulative inference, as we have defined it, is unsound for such machines because in effect they make a distinction between "base beliefs," sentences that have been explicitly *TELL*ed to the machine, and "derived beliefs," those to which the machine assents as a result of the mechanism. Since the logic has only a single belief operator, which describes both base beliefs and derived beliefs, such machines lead to contradictions. In Chapter 5, we discuss the work of Haas, who defines a different form of simulative inference in which the mechanism need not satisfy the closure constraint. This is possible because in his logic, each belief attribution is indexed with an upper bound on the time at which the agent came to hold that belief. In one sense, our form of simulative inference is less general than that of Haas since it is applicable for a smaller class of machines; but in another sense, it is more general, since to apply it one doesn't need as much information about the believer.

C2 (commutativity) *For any belief state S and acceptable sequence of sentences $\varphi_1, \ldots, \varphi_n$, and for any permutation ρ of the integers $1 \ldots n$, the sequence $\varphi_{\rho(1)}, \ldots, \varphi_{\rho(n)}$ is also acceptable, and*

$$\mathcal{B} \cdot \mathit{TELL}(S, \varphi_1, \ldots, \varphi_n) = \mathcal{B} \cdot \mathit{TELL}(S, \varphi_{\rho(1)}, \ldots, \varphi_{\rho(n)}).$$

The commutativity constraint says that if a sequence of sentences is acceptable, then it is acceptable in any order, and the belief set of the resulting state does not depend on the order.

It is clear that some form of commutativity is necessary for our simulative inference rule to be sound, since a pair of premises $B(a, \varphi)$ and $B(a, \psi)$ gives no information about which of φ and ψ came to be believed first. However, the constraint we have stated here stops short of requiring commutativity for all sequences of *TELL*s. It permits the belief machine to take order into account when deciding how to handle input sequences that are not acceptable. Typically these would be sequences in which the machine detects a contradiction.

C3 (monotonicity) *If a sequence $\varphi_1, \ldots, \varphi_n$ is acceptable, then it is monotonically acceptable.*

This constraint requires that if a *TELL* causes the retraction of some previously held beliefs, then some previously *TELL*ed sentence must be among the retracted beliefs.

Suppose a machine, having been *TELL*ed φ, assents to ψ by default unless it can prove χ. The rule of simulative inference is not sound for such a machine: it licenses the conclusion $B(a, \psi)$ from the premise $B(a, \varphi)$, but the conclusion is not entailed by the premise, since there is a belief state in which φ is believed but ψ is not, namely $\mathit{TELL}(S_0, \varphi, \chi)$. The monotonicity constraint rules out such a machine, since the sequence φ, χ is acceptable but not monotonically acceptable (ψ is believed after the first *TELL*, and no longer believed after the second).

The monotonicity constraint does not completely eliminate the possibility of retraction of beliefs. While it rules out defeasible inference, it does permit machines that, when they discover that their input contains a contradiction, choose some part of the input to ignore.

C4 (acceptable basis) *For any belief state S, there exists an acceptable sequence of sentences $\varphi_1, \ldots, \varphi_n$ such that*

$$\mathcal{B} \cdot \mathit{TELL}(S_0, \varphi_1, \ldots, \varphi_n) = \mathcal{B} \cdot S.$$

The acceptable basis constraint says that for each belief state, a state with the same belief set can be reached from the initial state by *TELL*ing the machine a finite, acceptable sequence of sentences.

The monotonicity constraint stated that when retraction occurs, a *TELL*ed sentence must be among the retracted beliefs; the acceptable basis constraint further requires that the effect of the retraction must be the same as that of ignoring part of the input. Note that the effect is not necessarily that of ignoring one of the input *sentences*—for example, the sequence $\varphi \wedge \psi, \neg\psi$ might induce the belief set $\{\varphi\}$.

In the next chapter, we will show that the simulative inference rule is sound given any belief machine that satisfies the above constraints. The constraints are sufficient, but not necessary, for the soundness of simulative inference. In particular, there are machines which violate the acceptable basis constraint, but for which simulative inference is sound. We use this particular form of the acceptable basis constraint because it is particularly natural and easily verified.

2.3 Negative Simulative Inference

For completeness, we will also need another form of simulative inference, one that allows us to detect by simulation that a set of sentences is not simultaneously believable. As in the former rule, the φ_i in this rule must be sentences, not open formulas; α may be any term.

$$\frac{B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)}{\bot}$$

if *ASK*(*TELL*$(S_0, \varphi_1, \ldots, \varphi_n), \varphi_i) = no$ for some $i, 1 \leq i \leq n$.

This rule says that if *TELL*ing a set of sentences to a belief machine in its initial state doesn't cause it to believe all of those sentences, then no agent can believe all of them simultaneously. In the next chapter, we will show that it is sound for any belief machine that satisfies the closure, commutativity, and acceptable basis constraints.

2.4 Indexicality and Introspection

A believer has the property of positive introspection if, for each sentence φ that he believes, he also believes that he believes φ; negative introspection means that when the agent doesn't believe φ, he believes that he doesn't believe φ.

Our model of belief seems essentially compatible with introspection, since one could build an introspective belief machine as follows: when queried about the sentence $B(\alpha, \varphi)$, where α is a term the agent uses to refer to itself, the machine could simply query itself about the sentence φ, and answer yes if the answer to the sub-query is yes (similarly for negative introspection). Unfortunately, this intuition is not easily realized in the model as we have defined it so far. In order to perform introspection, an agent's belief machine must be able to recognize terms that refer to that agent; but in general, a belief machine can't be said to know anything about the denotation of terms. It simply manipulates symbols. (In Section 2.5.3 we look more closely at the question of what an agent should be considered to know about the meaning of expressions in its language of thought.) One exception is that if a belief machine has "hard-coded" beliefs or inferential tendencies involving a particular symbol, then it can be considered to incorporate information about the preferred interpretation of that symbol. For example, one could construct a belief machine for which numerals are special symbols, about which it has predetermined beliefs, and which it can use in particular kinds of symbol manipulations (*i.e.* numerical calculations). Similarly, intuition says that a machine could have a particular constant, say me, which it recognizes as referring to the agent of which it is a part, and which therefore triggers introspective inference.

For the purpose of simulative inference, we have made the assumption that all agents have the same belief machine, and therefore if the machine has a special constant it treats as denoting the agent doing the reasoning, then the denotation of that term

should not be taken to be fixed by a model alone, but should depend on the context in which it is used. In other words, it is an indexical constant; and the semantics we have used so far does not support indexicality.

An alternative, which avoids the complication of indexicality in the semantics, would be to say that each agent uses a different "ego constant" to refer to itself, and to make the ego constant a modifiable parameter of the belief machine. Then the standard denotational semantics would be sufficient, since each ego constant would denote only one reasoner. However, this scheme arguably makes the syntax more difficult to understand. For instance, the meaning of $B(a, P(a))$ would depend on whether a were an ego constant. If so, then the sentence would be a report of a belief a has about himself; if not, then it would report a belief that a has involving the name a, but would contain no information about whether a knows that the name a refers to him (again, see Section 2.5.3 on the matter of what an agent should be said to know about the meaning of expressions that occur in its beliefs). We find that using the indexical results in a language whose meaning is more intuitively clear.

For the semantics of me, we use a standard technique from linguistic semantics: the denotation of an expression is in general no longer determined by a model alone, but by a model and an individual from the domain of that model, the latter being the reasoner from whose point of view the expression is considered. Given a model $M = \langle D, I, \gamma \rangle$ and a reasoner $r \in D$, the denotation of ground terms are determined as before, except that $|me|^{M,r} = r$.

In the semantics of quantifying-in, we want to ensure that

$$B(a, B(b, P(me)))$$

entails

$$\exists x[x = b \wedge B(a, B(b, P(x)))],$$

and does not entail

$$\exists x[x = a \wedge B(a, B(b, P(x)))].$$

To this end, we extend the concept of a variable substitution that is extension-preserving relative to a model, to that of a variable substitution that is extension-preserving *in a formula*, relative to a model and a reasoner. A variable substitution σ is extension-preserving in formula φ relative to model M and reasoner r if

1. for each variable ν that occurs at the top level of φ, *i.e.* outside of belief contexts, it is the case that $|\nu|^{M,r} = |\sigma(\nu)|^{M,r}$, and

2. for every belief atom $B(\alpha, \psi)$ at the top level of φ, σ is extension-preserving (defined recursively) in ψ under M and $|\alpha|^{M,r}$.

The corresponding change in the semantics of B should be clear: a belief atom $B(\alpha, \varphi)$ is now true in M relative to r iff there exists some variable substitution σ which is extension-preserving in φ relative to M and r such that $\mathit{ASK}(\gamma(|\alpha|^{M,r}), \varphi^{\sigma}) = yes$.

The indexical constant me allows the construction of introspective belief machines, as well as an axiomatic description of introspective believers. We will explore these possibilities in Section 3.5, and in Section 3.6 we will discuss simulative inference using introspective belief machines. Of particular interest is negative introspection: negative introspection is a kind of nonmonotonicity, and therefore invalidates the positive simulative inference rule; but for a belief machine with negative introspection, the negative simulative inference rule, which does not require monotonicity for its soundness, becomes more powerful.

Because of the extra complexity the indexical introduces, in the remainder of this dissertation we will use the language as originally defined, without the indexical, except where it is specifically needed. Many of the technical results will still hold if the indexical is introduced, but we will point out a few places where it is problematic.

2.5 Philosophical Considerations

In the next chapter, we will prove that our logic has certain mathematical properties, including the property of soundness and a form of completeness for the simulative inference rules given a belief machine that satisfies the constraints introduced above. These properties are matters of mathematical truth, and are not subject to debate except by the discovery of flaws in the proofs. However, there are less precisely answerable questions about what these abstract mathematical facts tell us about the world, and what consequences they have for the implementation of reasoning about belief in an AI system. We need to ask whether the belief machine abstraction accurately models some observable or inferable feature of real humans and their beliefs, and whether the constraints under which we have proved simulative inference sound permit the kinds of inference that real believers perform.

2.5.1 The Nature of the Objects of Belief, and the Semantics of Belief Reports in Natural Language

Belief can be viewed as a relation which holds between a person and some other kind of thing. The nature of that other thing has been a matter of much discussion. The proposals can be divided into two main groups: some hold that it is a sentence, *i.e.* a string of symbols in some language, and others that it is a proposition, the sort of thing that sentences express.

Among propositional theories of belief, there is a range of theories of the proposition, which vary in the fineness of the distinctions they make. In traditional Tarskian semantics, the meaning of a sentence is nothing but its truth value, so there are essentially only two propositions, the true one and the false one. This is clearly insufficient

to distinguish the objects of belief, and logicians have proposed a number of more fine-grained alternatives, using two techniques: intensions and structured propositions.

The idea that an expression has not only a denotation but a "sense" originates with Frege [1892]. Montague [1973] gave a model-theoretic definition of sense in a possible worlds framework, defining the sense of a sentence to be the set of worlds in which it is true. This makes it possible for sentences with the same truth value to express different propositions, but still has the consequence that *necessarily* equivalent sentences express the same proposition. In situation semantics [Barwise and Perry, 1983; Hwang and Schubert, 1993], the model is further refined, distinguishing propositions that are logically equivalent but involve different atomic predications. Even this level of granularity is arguably not fine enough. For example, Moore and Hendrix [1982, p. 96 in [Moore, 1995]] suggest that one can believe that door A is locked whenever door B is not, without believing that door B is locked whenever door A is not, but these two sentences have the same intension even in the most fine-grained intensional semantics.

In the Russellian view, a proposition is a structured entity. The meaning of a sentence is a structure whose constituents are the meanings of the constituents of the sentence. In other words, some of the syntactic structure of a sentence is visible in the structure of the proposition that it expresses.

Both the intensional and the structured forms of propositions cause problems with belief reports involving proper names, such as "Lois Lane does not believe that Clark Kent can fly." According to an argument of Kripke's [1980] which seems to be quite widely accepted, in a possible worlds model proper names must be "rigid designators," *i.e.* must denote the same thing in every world. If this is so, and "Clark Kent" and "Superman" refer to the same person in actuality, then they refer to the same person in all possible worlds, and therefore the proposition that Clark Kent can fly is the same as the proposition that Superman can fly. Consequently, if "Lois believes that Superman can fly" is true, then "Lois believes that Clark Kent can fly" must be true as well, even if Lois believes that the names "Clark Kent" and "Superman" refer to two different people. Similarly, several authors claim that the contribution of a proper name to a proposition is nothing but its denotation [Salmon, 1986; Soames, 1988; Braun, 1998], with the same unintuitive conclusion. We will later refer to this view as the "direct reference theory."

Of those who argue for sentences as the objects of belief, some hold that belief involves a disposition towards a sentence of a natural language [Carnap, 1947], while others hold that it involves a sentence of the agent's "language of thought" (LOT) [Fodor, 1975].[1] Sentential theories can be more fine-grained than any propositional theory yet proposed. They do not seem to have any difficulty with proper names: "Clark Kent can fly and "Superman can fly," regardless of what propositions they express, are clearly

[1] If there is in fact a language of thought, then literally speaking it is at least as "natural" as English; but for lack of a better term, we will continue to use the phrase "natural language" in the usual way, to mean a spoken or written language such as English.

not the same sentence. LOT theories also have a particular appeal for use in logical AI, since that entire endeavor is predicated on the idea that reasoning is the manipulation of symbols in a language of thought.

The simplest formulation of a sentential theory involving natural language would be that "α believes that φ" is true iff α is disposed to assent to φ. Complications arise for so-called *de re* belief reports, in which the speaker does not wish to convey anything about how the believer refers to some entity, for example, "Lois believes that Clark wears glasses," uttered in a context in which Lois is familiar with Clark but doesn't know his name; but such cases can be accommodated, probably to the satisfaction of most, by a sufficiently nuanced theory of quantifying-in [Kaplan, 1969]. A more difficult problem is that we can, in English, attribute beliefs to someone who does not understand English, and would therefore not be disposed to assent to any English sentence (this objection was apparently first voiced by Church [1950]). The intuitively attractive solution is to modify the semantics of belief as follows: "α believes that φ," uttered in English, is true iff α is disposed to assent to some sentence φ' such that φ' is a translation into a language that α understands of the English sentence φ. But this requires that we define what makes a sentence of one language a translation of a sentence in another language. The most plausible condition for φ' being a translation of φ would seem to be that they both express the same proposition, but this simply reintroduces the problematic question of the nature of propositions, from which we had hoped sentential theories would rescue us.

The same problem of translation arises for theories that involve sentences of a language of thought, rather than a natural language (such as the theory presented in this dissertation). Such theories say that "α believes that φ" is true iff there is a sentence φ' which is a translation into α's LOT of the English sentence φ, such that α stands in a particular relationship to φ'. This relationship might be, for example, having a token of φ' stored in an appropriate way in α's mind, or, in the case of our theory, α's belief machine being in a state such that it will assent to φ'.

We will say a little more about the problem of identifying translations shortly, but we will not claim to have solved it. Let us examine exactly what it is that we have failed to do, in order to understand what we can still hope for. The question we have failed to answer is one of linguistic semantics, of the truth conditions of English sentences of the form "α believes that φ." This is an important question, both for semantics in general and for our research program in particular (see Chapter 4), but it is not the only interesting question involving belief. Of perhaps even more importance for AI is the problem of psychological modeling, of understanding and predicting behavior in terms of mental states. Beliefs are an essential component of naive psychology, and in this role it is both plausible and useful to consider them to be mental representations, regardless of one's convictions about the nature of propositions and objective truth.

In fact, if there is anything in the debate about belief on which a consensus appears to be emerging, it is the idea that a complete theory of belief must somehow include

mental representations. The advocates of this idea range from Rapaport *et al.* [1997] and Jackendoff [1983], who entirely eliminate denotation and truth from their semantics, maintaining that the meaning of a natural language sentence is nothing but the mental representation to which it gives rise, to Russellian advocates of the direct reference theory such as Salmon [1986], Soames [1988], and Braun [1998], as well as Fregeans such as Cresswell [1985], who hold that belief is a relationship that a believer has to a proposition, but that the relationship is mediated by a mental representation, which is a "way of believing" the proposition.

Since we do not hope to resolve the controversy over the semantics of belief reports, we can remain noncommittal about the relationship between a sentence $B(\alpha, \varphi)$ of our logic and the English sentence "α believes that φ." (We may occasionally be caught paraphrasing the former using the latter; this should be seen as an intuitive aid only, not a statement of equivalence. The logical sentence does make assertions about the believer's way of referring to the entities involved in the content of the belief, while it is a matter of controversy whether the English sentence does that. Even an advocate of the direct reference theory will agree that to a naive reader, the English sentence gives rise to intuitions that match the intended meaning of the logical sentence.)

2.5.2 The Assumption of Identical Belief Machines

For the purposes of studying simulative inference, we make the essential assumption that all believers have functionally identical belief machines, *i.e.* that if two agents have had the same experiences, then they will have the same beliefs. This assumption is impossible to verify or falsify in practice, since no two people have completely identical experiences from birth, but the idea does raise some intuitive resistance. That is, it seems possible that some people are more intelligent than others by virtue of genetic traits, and that different people have different kinds of insight because of differences in the structure of their brains. Nevertheless, if humans reason successfully about belief by simulation, and we believe they do, then there must be some inferential ability that is common, and known to be common, to all. Of particular interest to us is the reasoning that people do when understanding language. It is widely accepted that much of the information conveyed by a linguistic utterance is implicit—speakers rely on their hearers to infer a great deal beyond what is explicitly present in the utterance. This would seem to require that the speaker have certain assumptions about the hearer's inferential ability, and simulative inference seems a reasonable way for these assumptions to be incorporated into the reasoning process.

Ultimately, there may be no practical difference between allowing believers to have different inferential ability, and assuming they have identical inferential ability but allowing them to have arbitrarily different experiential histories. In practice we can never have complete knowledge of another agent's experience, and the model permits agents

to learn new inference methods from experience, so the assumption that all agents have the same inferential ability can be a rather weak one.

We will prove the soundness and (in a restricted sense) completeness of simulative inference, under the assumption that the belief machine of the agent being reasoned about is the same as that used for the simulation. But for the foreseeable future, when a computer system simulates the beliefs of a human, it will not be using a belief machine of the same inferential strength. Since the assumption is not valid, the soundness and completeness results do not strictly hold. However, they can still have some approximate predictive value for the real world, to the extent that the simulation algorithm approximates human reasoning. The soundness result is likely to fare better than the completeness result, since an algorithm for simulating human beliefs is likely to believe less, not more, than an actual human given the same initial information. That is, the conclusions drawn by simulative inference are likely to be correct, but some conclusions that could be drawn with a better simulation are likely to be missed.

2.5.3 The Assumption of a Shared Language of Thought

Since we define the belief machine as a computational mechanism that manipulates sentences of the language of thought, and we assume that all agents share a common belief machine, we implicitly assume that all agents use the same language of thought (using 'language' in the restricted technical sense, meaning a set of sentences defined by a set of symbols and syntactic rules for combining them). Furthermore, for belief attributions to have the intended meaning, we must also assume that all agents use the symbols of the language in the same way. We have defined the logic such that the assertion $B(John, Tall(Mary))$ necessarily means that the belief machine of the agent denoted by $John$ assents to the sentence $Tall(Mary)$, but that doesn't mean that he believes Mary to be tall, unless we also assume that he uses the symbol $Tall$ to represent the property of being tall, and the symbol $Mary$ to represent Mary.

It seems reasonable enough to assume that the syntax of the language of thought is hard-wired in the human mind, but perhaps less reasonable that there is a single shared set of symbols and associated meanings. We can make the assumption slightly more palatable by using the following essentially equivalent form: that each believer has his own set of symbols, but for each symbol in one believer's idiolect, there is a unique corresponding symbol in each other's idiolect. This brings us back to the problem of translation, which we first faced in Section 2.5.1.

Note that shared denotation is not a sufficient criterion for identifying corresponding symbols in different idiolects. Two symbols of the same idiolect might happen to denote the same individual without the believer knowing it; in such a case, the two may not be equally appropriate candidates for the translation of a coextensive term of another idiolect. Consider two agents, Lois and Jimmy, and let us represent symbols of Lois' idiolect with the subscript l and those of Jimmy's idiolect with the subscript j. If

Lois' language of thought has two symbols $Superman_l$ and $Kent_l$, and her belief machine assents to the sentence $\neg(Superman_l = Kent_l)$, then $flies_l(Superman_l)$ and $flies_l(Kent_l)$ are not equally appropriate representations in her idiolect of Jimmy's sentence $flies_j(Superman_j)$.

This indicates two problems for a full theory of translation. The first is that, since our model theory assigns a symbol no meaning beyond its denotation, we would need to enrich the model theory in order to define the translation relationship fully. But second, even given a (perhaps intensional) model theory that provides a fine enough individuation of symbol meanings, there is a question of what a symbol's meaning, as given by a model, has to do with how the agent uses the symbol. The intended connection between the two is clear: a symbol has the meaning it does precisely because of the way the agent uses the symbol in its beliefs—because of the history of experiences that led the agent to form the concept of which the symbol is a physical manifestation. But we have given an agent's experiential history no place in the model theory. We have pointed out the desired condition that an agent may act as if two symbols refer to different things, even if they are in fact coreferential. But by the same token, in an intensional version of our model theory, an agent could act as if two symbols refer to different things, even if they had the same intension. Such a possibility violates intuitions about the notion of intension, yet an intensional version of our theory would permit it.

It is not clear to us how damaging this fact is to the theory. The possibility of models other than the intended one is certainly not a problem: the same possibility exists in classical logic, and as in classical logic, we can eliminate unintended models by adding axioms. For example, we might use an axiom stating that if someone believes something, then he will act as if it were true. But the undesired models we have been discussing are objectionable not merely because they support the truth of contingently false propositions, but because they seem to be inherently self-contradictory, given our intuitive understanding not only of the symbols of the logic, but of the components of the model theory itself. Nevertheless, if the undesired models can be eliminated by adding judiciously chosen axioms, there may be no adverse consequences for the proof theory.

2.5.4 The Constraints

When evaluating whether constraints C1–C4 are reasonable, *i.e.* whether the mechanisms by which humans maintain their beliefs can be said to satisfy these constraints, it is important to note that the belief machine need not be seen as describing an agent's entire reasoning capability. In particular, the set of sentences to which the belief machine answers yes need not be the same as the set of sentences to which the *agent* would assent if asked. The belief machine might be a single component of a larger reasoning system, perhaps best viewed as the information storage and retrieval component. The

belief machine describes an agent's current beliefs at a moment, and how those beliefs change in response to new information; it does not describe the set of things the agent could infer given an opportunity for reflection (the latter is closer to the idealized notion of belief inherent in the classical possible worlds model). In this light, it is clear that the closure constraint does not preclude an agent whose beliefs expand as a result of his being asked a question. If φ is a sentence that an agent has not previously considered, and which he can't immediately and effortlessly see to be either true or false, then at the time of the asking of the question the agent believes neither φ nor $\neg\varphi$, *i.e.* the *ASK* function returns no for both queries. When an agent whose belief machine is in such a state is asked the question φ, after finding that it does not currently have any convictions on the subject, it might undertake more intensive and goal-directed reasoning to try and discover whether its truth or falsity follows from what it knows. This reasoning involves capabilities of the agent that are not described by the belief machine. If the agent eventually decides that φ is true, then it will come to believe φ; this is modeled by the *TELL*ing of φ to the belief machine. As a result of the *TELL*, the agent's beliefs are augmented by φ and sentences that follow effortlessly from φ (combined with the existing beliefs), and perhaps with the retraction of some previously-held beliefs.

An Example Belief Machine

Besides the question of whether human belief can be said to satisfy the constraints we have listed, there is a related question of whether any interesting inference algorithms of the kind studied in AI satisfy the constraints. We will now describe a simple example of a non-trivial algorithm which satisfies the constraints. Our example machine does limited deductive reasoning by checking for clause subsumption, and it is able to revise its beliefs when given information that contradicts earlier inputs.

The machine's state consists of a list of believed clauses, maintained in the order in which they were learned. The *ASK* function works as follows. First, if its sentential argument contains any quantifying-in, it simply answers no. Otherwise, it converts the sentence to clause form, both at the top level and in belief contexts, distributing B over conjuncts. For example, the sentence

$$\neg B(a, P(c) \wedge P(d))$$

becomes

$$\neg[B(a, P(c)) \wedge B(a, P(d))]$$

which is further converted to

$$\neg B(a, P(c)) \vee \neg B(a, P(d)).$$

Note that, since quantifying-in is prohibited, a free variable occurring in a belief context in a clause can unambiguously be read as being bound by an implicit quantifier lying

within the narrowest containing belief context. That is, the clause $B(a, B(b, P(x)))$ is equivalent to $B(a, B(b, \forall x P(x)))$, not $B(a, \forall x B(b, P(x)))$ or $\forall x B(a, B(b, P(x)))$. Once the input sentence has been converted to a set of clauses, each clause is compared against the list of stored clauses that comprises the machine's state. If every clause of the query is subsumed by some stored clause, the function answers yes; otherwise, it answers no.

The *TELL* function, when given a sentence, first checks whether the sentence contains any quantified-in belief atoms or any positively embedded existential quantifiers. If so, it does nothing—that is, having rejected the input, the machine remains in the same state. If not, it next runs the *ASK* function on the sentence. If the answer is yes, then it does nothing—the sentence is already believed, so the machine remains in the same state. If the answer is no, it then runs the *ASK* function on the negation of the input sentence, to see if the input contradicts what is currently believed. If the answer to that query is no, then the machine enters a new state by adding the clause form of the original sentence to the list of believed clauses. If the answer is yes, indicating a contradiction, then the contradiction is resolved by removing some clauses, chosen as follows, from the list. Each clause in the negation of the input is subsumed by one or more clauses in the list. The clause from the negated input whose most recently learned subsuming clause is the earliest is the one chosen to be rejected. All clauses on the list that subsume that clause are discarded, so that the negation of the input is no longer believed. Then the input is added as above.

Note first of all that the *ASK* and *TELL* algorithms halt on all inputs (as long as the list of believed clauses is finite, and the list must be finite if there have been only finitely many preceding *TELL*s), and therefore they do define a belief machine. Furthermore, the machine satisfies constraints C1–C3, as we will now show, and therefore simulative inference using this machine is sound. The closure constraint is satisfied because before changing the belief state, *TELL* first calls *ASK*, and it remains in the same state if the proffered sentence is already believed. The acceptable basis constraint is satisfied because a belief state is simply a list of clauses, and that list of clauses itself, translated back into sentential form, is a monotonically acceptable sequence of sentences that could be *TELL*ed to a machine, starting from the initial state, to induce the same belief set. The order of *TELL*ed sentences is taken into account when revising beliefs, but this does not violate the commutativity constraint. This last constraint is satisfied because if a sequence of sentences is monotonically acceptable in S_0, then in the state resulting from *TELL*ing that sequence, the list of believed clauses is simply the list of input sentences converted to clause form; and the *ASK* function pays no attention to the order of the clause list, so the belief set is the same regardless of the order of the inputs.

3 Mathematical Properties of the Logic

Having defined our model of belief and explained the intuitions behind it, we now demonstrate some of its interesting mathematical properties. In Section 3.1, we prove that the four constraints discussed in Section 2.2 are sufficient for the soundness of the positive and negative simulative inference rules. In Section 3.2, we address the question of whether the constraints are also necessary for the soundness of the rules (some of them are, others are not). In Section 3.3, we introduce some general-purpose inference rules (as opposed to rules for reasoning specifically about belief) and prove their soundness. Then in Section 3.4, we address the matter of completeness. We show that no complete set of inference rules can exist for our logic. However, we also show that the set of inference rules introduced earlier, including the simulative rules, is complete for a syntactically restricted subset of the logic, given any belief machine for which the rules are sound. In Section 3.5 we list some axioms that, in various combinations, have been taken to be characteristic of belief in the classical literature, and we consider the constraints on the belief machine under which each of them is valid.

3.1 Soundness Proofs

Theorem 1 (Soundness of Simulative Inference) *For belief machine*

$$m = \langle \Gamma, S_0, \textit{TELL}, \textit{ASK} \rangle$$

satisfying constraints C1–C4, m-model $M = \langle D, I, \gamma \rangle$, and sentences $\varphi_1, \ldots, \varphi_n$ and ψ, if $M \models \bigcup_i^n \{B(\alpha, \varphi_i)\}$, and ASK(TELL$(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$, then $M \models B(\alpha, \psi)$.

The proof can be summarized as follows: by the acceptable basis constraint, agent α's current belief set can be induced by *TELL*ing the belief machine some sequence of sentences $\chi_1, \ldots, \chi_m$ which is acceptable, and therefore monotonically acceptable

thanks to the monotonicity constraint, in S_0. Let S be the state *TELL*$(S_0, \chi_1, \ldots, \chi_m)$. By the closure constraint, the sequence $\varphi_1, \ldots, \varphi_n$ is monotonically acceptable in S. This means that the whole sequence $\chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n$ is monotonically acceptable in S_0. By the definition of 'monotonically acceptable' and the commutativity constraint,

$$\mathcal{B} \cdot TELL(S_0, \varphi_1, \ldots, \varphi_n) \subseteq \mathcal{B} \cdot TELL(S_0, \chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n).$$

From the premises, we know that ψ is in the former belief set, so it must be in the latter as well. By the closure constraint, the latter is the same as $\mathcal{B} \cdot S$, so α believes ψ.

Proof: Assume that

$$M \models B(\alpha, \varphi_i) \tag{3.1}$$

for all $1 \leq i \leq n$, and assume that

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes. \tag{3.2}$$

Let $S = \gamma(|\alpha|^M)$, *i.e.* the belief state of the agent denoted by α. From (3.1) and the semantics of the operator B, it follows that for each φ_i, there is an extension-preserving variable substitution σ_i such that

$$\textit{ASK}(S, \varphi_i^{\sigma_i}) = yes. \tag{3.3}$$

Furthermore, since the rule only applies when the φ_i have no free variables, φ_i^{σ} is the same as φ_i for any σ. Therefore, (3.3) is equivalent to

$$\textit{ASK}(S, \varphi_i) = yes, 1 \leq i \leq n. \tag{3.4}$$

According to the acceptable basis constraint (C4), there is some sequence of sentences $\chi_1, \ldots, \chi_m$ which is acceptable in S_0, such that

$$\mathcal{B} \cdot \textit{TELL}(S_0, \chi_1, \ldots, \chi_m) = \mathcal{B} \cdot S. \tag{3.5}$$

Since the sequence is acceptable in S_0, by the monotonicity constraint (C3) it is also monotonically acceptable in S_0.

From (3.4) and (3.5), it follows that

$$\textit{ASK}(\textit{TELL}(S_0, \chi_1, \ldots, \chi_m), \varphi_i) = yes, 1 \leq i \leq n. \tag{3.6}$$

From (3.6) and n successive applications of the closure constraint (C1), it follows that

$$\mathcal{B} \cdot \textit{TELL}(S_0, \chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n) = \mathcal{B} \cdot \textit{TELL}(S_0, \chi_1, \ldots, \chi_m), \tag{3.7}$$

and that the sequence $\chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n$ is monotonically acceptable in S_0. By the commutativity constraint, the sequence $\varphi_1, \ldots, \varphi_n, \chi_1, \ldots, \chi_m$ is also acceptable in S_0, and therefore monotonically acceptable by the monotonicity constraint, and

$$\begin{aligned}\mathcal{B} \cdot \textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n, \chi_1, \ldots, \chi_m) &= \mathcal{B} \cdot \textit{TELL}(S_0, \chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n) \\ &= \mathcal{B} \cdot \textit{TELL}(S_0, \chi_1, \ldots, \chi_m) \quad \text{(see (3.7))} \\ &= \mathcal{B} \cdot S \quad \text{(see (3.5)).} \end{aligned} \tag{3.8}$$

Since the sequence $\varphi_1, \ldots, \varphi_n, \chi_1, \ldots, \chi_m$ is monotonically acceptable in S_0, by the definition of 'monotonically acceptable' it is also true that the sequence $\chi_1, \ldots, \chi_m$ is monotonically acceptable in $\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n)$, and hence

$$\mathcal{B} \cdot \textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n) \subseteq \mathcal{B} \cdot \textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n, \chi_1, \ldots, \chi_m) \tag{3.9}$$

Equation (3.2) says that

$$\psi \in \mathcal{B} \cdot \textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n) \tag{3.10}$$

From (3.8), (3.9), and (3.10), it follows that

$$\psi \in \mathcal{B} \cdot S \tag{3.11}$$

or, equivalently,

$$\textit{ASK}(S, \psi) = yes \tag{3.12}$$

Since ψ was assumed to have no free variables, ψ^σ is the same as ψ for any variable substitution σ. By the semantics of B and our choice of $S = \gamma(|\alpha|^M)$, the desired conclusion follows:

$$M \models B(\alpha, \psi) \tag{3.13}$$

□

Theorem 2 (Soundness of Negative Simulative Inference) *Given a belief machine that satisfies the closure, commutativity, and acceptable basis constraints, if*

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \varphi_i) = no$$

for some $1 \leq i \leq n$*, then* $\{B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)\}$ *is unsatisfiable.*

Proof: We will prove the contrapositive of the above statement. Assume that

$$\{B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)\}$$

is satisfiable, *i.e.* that there exists some state S such that $\mathit{ASK}(S, \varphi_i) = yes$ for all $1 \leq i \leq n$. By the acceptable basis constraint, there is some sequence of sentences $\psi_1, \ldots, \psi_m$ that is acceptable in S_0, such that

$$\mathcal{B} \cdot \mathit{TELL}(S_0, \psi_1, \ldots, \psi_m) = \mathcal{B} \cdot S.$$

This means that

$$\mathit{ASK}(\mathit{TELL}(S_0, \psi_1, \ldots, \psi_m), \varphi_i) = yes, 1 \leq i \leq n.$$

By the closure constraint, if we take a machine in state $\mathit{TELL}(S_0, \psi_1, \ldots, \psi_m)$ and *TELL* it each of the φ_i in turn, the resulting states will all have the same belief set, which means that the entire sequence

$$\psi_1, \ldots, \psi_m, \varphi_1, \ldots, \varphi_n$$

is acceptable in S_0. Therefore, the commutativity constraint applies; it says that the sequence

$$\varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_m$$

is acceptable in S_0, which by the definition of acceptability means that the initial subsequence $\varphi_1, \ldots, \varphi_n$ is acceptable in S_0. Therefore, all of the φ_i are elements of $\mathcal{B} \cdot \mathit{TELL}(S_0, \varphi_1, \ldots, \varphi_n)$. □

3.2 Are the Constraints Necessary?

We have shown that constraints C1–C4 are sufficient for the soundness of positive and negative simulative inference. The question naturally arises whether these constraints are also necessary for the soundness of the inference rules. In fact, they are not all necessary. The constraints we have used were chosen because they are particularly natural to express and easy to verify of a given inference algorithm, but some of them could be weakened while maintaining the soundness properties.

We begin with the most straightforward result: that the monotonicity constraint is necessary for the soundness of (positive) simulative inference.

Theorem 3 *If the belief machine violates the monotonicity constraint, then the positive simulative inference rule is not sound.*

Proof: If the monotonicity constraint is violated, then there is a sequence of sentences $\varphi_1, \ldots, \varphi_n$ which is acceptable, but not monotonically acceptable, in S_0. Since the sequence is not monotonically acceptable, there is some i, $1 \leq i \leq n$, and some sentence ψ, such that $\mathit{ASK}(\mathit{TELL}(S_0, \varphi_1, \ldots, \varphi_{i-1}), \psi) = yes$ but

ASK(*TELL*$(S_0, \varphi_1, \ldots, \varphi_i), \psi) = no$. ψ is not one of $\varphi_1, \ldots, \varphi_i$, since that sequence is acceptable. Then the simulative inference rule licenses the conclusion $B(a, \psi)$ from the premises $B(a, \varphi_1), \ldots, B(a, \varphi_{i-1})$, but there are models in which the premises are true and the conclusion false, namely models that assign the belief state *TELL*$(S_0, \varphi_1, \ldots, \varphi_i)$ to the agent denoted by a. Therefore, the rule is not sound. □

The commutativity constraint is not necessary for the soundness of either simulative rule alone, but is necessary for them both to be sound.

Theorem 4 *If the belief machine violates the commutativity constraint, then the positive and negative simulative inference rules are not both sound.*

Proof: If the commutativity constraint is violated, then there is some sequence $\varphi_1, \ldots, \varphi_n$ which is acceptable in S_0, and some permutation ρ of the integers $1 \ldots n$, such that either the sequence $\varphi_{\rho(1)}, \ldots, \varphi_{\rho(n)}$ is not acceptable in S_0 or $\mathcal{B} \cdot$ *TELL*$(S_0, \varphi_{\rho(1)}, \ldots, \varphi_{\rho(n)}) \neq \mathcal{B} \cdot$ *TELL*$(S_0, \varphi_1, \ldots, \varphi_n)$.

If the sequence $\varphi_{\rho(1)}, \ldots, \varphi_{\rho(n)}$ is not acceptable, let $\varphi_{\rho(1)}, \ldots, \varphi_{\rho(j)}$ be the shortest initial subsequence that is not acceptable. Then there is some i, $1 \leq i \leq j$, such that

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_{\rho(1)}, \ldots, \varphi_{\rho(j)}), \varphi_{\rho(i)}) = no,$$

so the negative simulative inference rule licenses the conclusion $\perp$ from the premises

$$B(a, \varphi_{\rho(1)}), \ldots, B(a, \varphi_{\rho(j)}),$$

in that order. But there is a state in which all of $\varphi_{\rho(1)}, \ldots, \varphi_{\rho(j)}$ are believed, namely

$$\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n),$$

so the rule is unsound.

If the reordered sequence is acceptable in S_0 but the resulting belief set is different, let

$$S = \textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n),$$

and let

$$S' = \textit{TELL}(S_0, \varphi_{\rho(1)}, \ldots, \varphi_{\rho(n)}).$$

This case can be further divided into two cases: there is some sentence ψ such that either *ASK*$(S, \psi) = yes$ but *ASK*$(S', \psi) = no$, or vice versa. In the first case, the positive simulative inference rule licenses the conclusion $B(a, \psi)$ from the premises $B(a, \varphi_1), \ldots, B(a, \varphi_n)$, used in that order; but there is a model in which $B(a, \varphi_1), \ldots, B(a, \varphi_n)$ are all true and $B(a, \psi)$ is false, namely one which

assigns the belief state S' to the agent denoted by a, so the rule is unsound. Similarly, in the second case the rule licenses the conclusion $B(a, \psi)$ from the premises $B(a, \varphi_{\rho(1)}), \ldots, B(a, \varphi_{\rho(n)})$ in that order, but in a model where a's belief state is S, the premises are all true and the conclusion is false. □

The closure constraint as stated in the previous chapter is stronger than is strictly needed. However, a weakened (but more complicated to state) version of the constraint can take the place of the original one in the soundness proofs, and is necessary for the joint soundness of the two simulative inference rules. The original constraint required that for any belief state S and sentence φ, if $\mathit{ASK}(S, \varphi) = yes$ then $\mathcal{B} \cdot \mathit{TELL}(S, \varphi) = \mathcal{B} \cdot S$. We can weaken the constraint by having S range only over states reachable via an acceptable sequence, *i.e.* states S for which $S = \mathit{TELL}(S_0, \varphi_1, \ldots, \varphi_n)$ for some sequence $\varphi_1, \ldots, \varphi_n$ which is acceptable in S_0.

Theorem 5 *If the belief machine violates the weakened closure constraint, then the positive and negative simulative inference rules are not both sound.*

Proof: If the weakened closure constraint is violated, then there is a sequence $\chi_1, \ldots, \chi_n$ which is acceptable in S_0, a state $S = \mathit{TELL}(S_0, \chi_1, \ldots, \chi_n)$, a sentence $\varphi \in \mathcal{B} \cdot S$, and a sentence ψ such that either

- $\mathit{ASK}(S, \psi) = no$ and $\mathit{ASK}(\mathit{TELL}(S, \varphi), \psi) = yes$, or
- $\mathit{ASK}(S, \psi) = yes$ and $\mathit{ASK}(\mathit{TELL}(S, \varphi), \psi) = no$.

In the first case, the sentence φ is an element of $\mathcal{B} \cdot S$, as are $\chi_1, \ldots, \chi_n$, but ψ is not. Therefore, there is a model in which $B(a, \chi_1), \ldots, B(a, \chi_n), B(a, \varphi)$ are all true but $B(a, \psi)$ is false, namely one that assigns the belief state S to the agent denoted by a. But the positive simulative inference rule licenses the conclusion $B(a, \psi)$ from the premises $B(a, \chi_1), \ldots, B(a, \chi_n), B(a, \varphi)$, so it is unsound.

In the second case, either ψ is one of $\chi_1, \ldots, \chi_n, \varphi$, or it isn't. If it is, then the negative simulative inference rule is unsound: $\mathit{ASK}(\mathit{TELL}(S_0, \chi_1, \ldots, \chi_n, \varphi), \psi) = no$ for some $\psi \in \{\chi_1, \ldots, \chi_n, \varphi\}$, so the rule licenses the conclusion $\perp$ from the premises $B(a, \chi_1), \ldots, B(a, \chi_n), B(a, \varphi)$, but there is a model in which those premises are all true, namely one that assigns the belief state S to the agent denoted by a. Otherwise, the monotonicity constraint is violated, because the sequence $\chi_1, \ldots, \chi_n, \varphi$ is acceptable but not monotonically acceptable, and by Theorem 3 it follows that the rule of simulative inference is unsound. □

The last remaining constraint is the acceptable basis constraint. It is not necessary for the soundness of the simulative inference rules; it is simply a very natural constraint which happens to be useful in the soundness proofs.

Theorem 6 *The acceptable basis constraint is not necessary for the soundness of positive or negative simulative inference.*

Proof: We will construct a belief machine with the following properties. The only sentences it accepts are atomic formulas with zero or more negations. The only inference it ever performs is the adding or stripping of even numbers of negations. Initially, it only answers yes to sentences that can be derived by stripping zero or more pairs of negations from something it has been *TELL*ed; but if it detects that it has been *TELL*ed sentences that contradict each other, it thenceforth also answers yes to sentences that can be obtained by adding one or more pairs of negations to sentences it has been *TELL*ed.

The machine's state consists of a list of sentences, initially empty, and a flag, initially unset, which indicates whether any beliefs have been retracted. When *TELL*ed a new sentence, it first adds it to the list. Then, if the sentence is of the form $\neg\neg\varphi$, it recursively *TELL*s itself φ. If it is of the form $\neg\varphi$ where φ is atomic, it checks the list for φ, and if it is present, it sets the flag and removes any sentences φ, $\neg\neg\varphi$, *etc.* that are present on the list. Similarly, if the *TELL*ed sentence is an atomic sentence φ , it checks the list for $\neg\varphi$, and if it is found, sets the flag and removes $\neg\varphi$, $\neg\neg\neg\varphi$, *etc.* from the list.

When *ASK*ed about a sentence, it answers yes if the sentence is on its list. If the sentence is not on the list, and the flag is unset, then it answers no. If the sentence is not on the list, but the flag is set and the sentence is of the form $\neg\neg\varphi$, it recursively *ASK*s itself φ, and returns the answer of the recursive query.

Let $\neg^n\varphi$ be an abbreviation for φ negated n times. Note that in any state, for any atomic sentence φ, if $\neg^{2n}\varphi$ is believed for some $n \geq 0$ then $\neg^{2i}\varphi$ is believed for every $0 \leq i \leq n$, and if $\neg^{2n+1}\varphi$ is believed for some $n \geq 0$ then $\neg^{2i+1}\varphi$ is believed for every $0 \leq i \leq n$. It follows that in any state and for any sentence φ, it is impossible that both φ with an odd number of negations and φ with an even number of negations are believed. When either one comes to be believed, if the other was previously believed, it is retracted.

This machine violates the acceptable basis constraint: the belief set of the state *TELL*$(S_0, \varphi, \neg\varphi, \varphi)$ is the infinite set $\{\varphi, \neg\neg\varphi, \neg\neg\neg\neg\varphi, \ldots\}$. There is no acceptable sequence that will induce this belief set, because the belief set is only infinite in states in which the flag is set, and the flag is only set in response to a non-acceptable input sequence.

Positive simulative inference is sound for this machine. For any set of premises $B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)$, if there is any model that satisfies all of the premises, then no two of $\varphi_1, \ldots, \varphi_n$ are contradictory, so the simulation's flag will not get set. Therefore, the only sentences to which the simulation will answer yes are those that can be derived from one of $\varphi_1, \ldots, \varphi_n$ by stripping even numbers of negations. As we have already observed, in every belief state in which the φ_i are believed, these conclusions must also be believed.

Negative simulative inference is sound as well. If $\mathit{ASK}(\mathit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \varphi_i) = no$ for some $1 \leq i \leq n$, then φ_i must be the complement of some φ_j, $i < j \leq n$; and in that case, $B(\alpha, \varphi_i)$ and $B(\alpha, \varphi_j)$ are not simultaneously satisfiable. □

3.3 Other Inference Rules

We now give the remainder of the set of inference rules that will be proved complete in Section 3.4. The rules are all variations of ones familiar from ordinary first-order logic, with the primary modification being restrictions on the contexts in which they can be applied.

These rules apply to all formulas, not just closed ones.

3.3.1 Rules of Propositional Logic

We adopt a complete set of standard inference rules for propositional logic. It will be convenient later on, in the completeness proof, to consider a formula of the form $\varphi \supset \psi$ as an abbreviation for $\neg\varphi \vee \psi$, so we will not bother with inference rules for the connective $\supset$.[1]

- Commutativity rules

$$\frac{\varphi \vee \psi}{\psi \vee \varphi} \qquad \frac{\varphi \wedge \psi}{\psi \wedge \varphi}$$

- $\wedge$-introduction, elimination

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \qquad \frac{\varphi \wedge \psi}{\varphi}$$

- $\vee$-introduction

$$\frac{\varphi}{\varphi \vee \psi}$$

- DeMorgan's rules

$$\frac{\neg(\varphi \wedge \psi)}{\neg\varphi \vee \neg\psi} \qquad \frac{\neg\varphi \vee \neg\psi}{\neg(\varphi \wedge \psi)} \qquad \frac{\neg(\varphi \vee \psi)}{\neg\varphi \wedge \neg\psi} \qquad \frac{\neg\varphi \wedge \neg\psi}{\neg(\varphi \vee \psi)}$$

[1] Other inference rules, such as double negation and disjunction elimination, can be derived from these rules. The particular set of rules presented here was chosen simply because it facilitates the completeness proof.

- Reductio ad absurdum

$$\dfrac{\begin{array}{c}\neg\varphi \\ \vdots \\ \bot\end{array}}{\varphi}$$

All but the last rule are trivially sound by the semantics of the connectives. The *reductio* rule, which justifies the conclusion φ when assuming $\neg\varphi$ leads to a contradiction, can be proved sound by induction on the length of the assumed proof.

3.3.2 Rules for Handling Equality

In ordinary FOL, substitution of one term for another is permitted if the two terms are known to denote the same thing. In our logic, such substitutions are only allowable outside of belief contexts. The restricted rule of substitution is:

$$\dfrac{\alpha = \beta \qquad \varphi}{\varphi'}$$

> where φ' is φ with the term β substituted for one or more occurrences of α *that are not in belief contexts*, such that none of the occurrences of α being replaced contain variables bound in φ, and such that the substitution does not cause any free variable of β to become bound in φ' .

Since it applies only in extensional (non-belief) contexts, this rule is trivially sound by the semantics of equality.

There is one situation in which substitution of a coextensive term is sound in belief contexts, which we cover with a separate rule, the rule of substitution in negative belief contexts:

$$\dfrac{\neg B(\alpha, \varphi) \qquad \nu = \tau}{\neg B(\alpha, \varphi_{\tau/\nu})}$$

> where ν is a free variable of φ, and substituting τ for ν in φ does not cause any free variable of τ to become bound.

Theorem 7 (Soundness of Substitution in Negative Belief Contexts) *For model M, formula $B(\alpha, \varphi)$, variable ν, and term τ such that the substitution of τ for ν in φ would not cause any free variable of τ to become bound, if $M \models \neg B(\alpha, \varphi)$ and $M \models \nu = \tau$ then $M \models \neg B(\alpha, \varphi_{\tau/\nu})$.*

Proof: Assume that $M \models \neg B(\alpha, \varphi)$, meaning that there is no variable substitution σ which is extension-preserving under M such that $M \models B(\alpha, \varphi^{\sigma})$. If

$M \not\models \neg B(\alpha, \varphi_{\tau/\nu})$, then there would have to be an extension-preserving substitution σ' such that $M \models B(\alpha, (\varphi_{\tau/\nu})^{\sigma'})$. If that were the case, we could construct another substitution σ which was identical to σ' except that $\sigma(\nu)$ was $\tau^{\sigma'}$. With this new σ, $M \models B(\alpha, \varphi^{\sigma})$, contrary to our assumption. □

This soundness proof depends on the original definition of the semantics, without indexicality (see Section 2.4). If the indexical constant me is allowed, and it occurs in the term τ, then the occurrence of me in the top-level equation $\nu = \tau$ doesn't represent the same thing as the occurrence embedded in the belief atom. Fixing the rule to handle indexicality is quite complicated, because occurrences of me in the belief atom could be contained within belief environments that are nested to an arbitrary depth.

We also need an identity axiom and a rule of commutativity:

$$\tau = \tau \qquad\qquad \frac{\tau_1 = \tau_2}{\tau_2 = \tau_1}$$

3.3.3 Restricted ∀-Instantiation

The instantiation of a universally-quantified variable with an arbitrary term is sound when it takes place in extensional contexts, just as in ordinary FOL, but is not always sound in belief contexts: it is possible for $\forall x B(a, P(x))$ to be true yet $B(a, P(c_1))$ false, for example if c_2 denotes the same thing as c_1 and $B(a, P(c_2))$ is true.

∀-instantiation is only unsound when the variable being instantiated occurs in a positively embedded belief context. To make the definition of a positive embedding precise, we treat $\exists \nu \psi$ as an abbreviation for $\neg \forall \nu \neg \psi$, and $\psi \supset \chi$ as an abbreviation for $\neg \psi \vee \chi$. With these definitions, an occurrence of a formula ψ within a formula φ is negatively embedded in φ if it lies in the scope of an odd number of of negations, and is positively embedded otherwise.

The rule is as follows, where $\varphi_{\tau/\nu}$ is the result of substituting term τ for all free occurrences of variable ν in φ:

$$\frac{\forall \nu \varphi}{\varphi_{\tau/\nu}}$$

> if the substitution of τ for ν in φ does not cause any free variable of τ to become bound, and no occurrence of ν that is free in φ is within a belief context that is positively embedded in φ.

This rule is only sound in the absence of the indexical. It could be made sound for the version of the language with an indexical me by prohibiting me from occurring in τ, but then the rule would no longer perform its role in the completeness proof (below). A version that is sound and makes the necessary contribution to completeness

undoubtedly exists, but it would significantly complicate the proofs, and we have not attempted to formulate one.

In proving the soundness of this inference rule, we will use the following lemma:

Lemma 8 *Let φ be any formula, ν a variable that occurs free in φ, τ a term whose substitution for ν in φ doesn't cause any free variable of τ to become bound, and $M_1 = \langle D, I_1, \gamma \rangle$ a model. Let M_2 be a model $\langle D, I_2, \gamma \rangle$, where D and γ are the same as in M_1, and I_2 is the same as I_1 except that $I_2(\nu)$ is $|\tau|^{M_1}$. Then*

(a) if no free occurrence of ν in φ is in a negatively embedded belief context, and $M_1 \models \varphi_{\tau/\nu}$, then $M_2 \models \varphi$;

(b) if no free occurrence of ν in φ is in a positively embedded belief context, and $M_2 \models \varphi$, then $M_1 \models \varphi_{\tau/\nu}$.

Note that the lemma entails that if no free occurrence of ν is in any belief context at all, then $M_2 \models \varphi$ if and only if $M_1 \models \varphi_{\tau/\nu}$.

Proof: The proof is by induction on the complexity of φ. The bases of the induction are the atomic formulas. The lemma clearly holds for ordinary (non-belief) atoms, and for belief atoms in which ν does not occur free in the second argument, because these are extensional contexts. For the case of a belief atom containing a free occurrence of ν in the belief context, assume that $M_1 \models B(\alpha_{\tau/\nu}, \psi_{\tau/\nu})$. This means that there is some variable substitution σ, which is extension-preserving under M_1, such that *ASK*$(\gamma(|\alpha_{\tau/\nu}|^{M_1}), (\psi_{\tau/\nu})^\sigma) = yes$. Construct a new variable substitution σ' identical to σ except that $\sigma'(\nu)$ is τ^σ. Then σ' is extension-preserving under M_2: it agrees with σ on every variable other than ν, therefore preserving the extension of those variables under M_2 (as well as M_1); and as for ν,

$$
\begin{array}{rcll}
|\sigma'(\nu)|^{M_2} & = & |\tau^\sigma|^{M_2} & \text{by the construction of } \sigma' \\
& = & |\tau^\sigma|^{M_1} & \text{because } M_1 \text{ and } M_2 \text{ agree on the interpretation of all ground terms} \\
& = & |\tau|^{M_1} & \text{since } \sigma \text{ is extension-preserving under } M_1 \\
& = & I_2(\nu) & \text{by the construction of } I_2.
\end{array}
$$

Then $\psi^{\sigma'}$ is $(\psi_{\tau/\nu})^\sigma$ (by the construction of σ'), and $|\alpha|^{M_2}$ is the same as $|\alpha_{\tau/\nu}|^{M_1}$ (by the construction of M_2), so *ASK*$(\gamma(|\alpha|^{M_2}), \psi^{\sigma'}) = yes$, so $M_2 \models B(\alpha, \psi)$. This proves that part (a) of the lemma holds for atomic formulas; part (b) holds trivially, because if an occurrence of ν in an atomic formula is not in a positively embedded belief context, then it is in no belief context at all.

For the induction step, we must show that if the lemma holds for two formulas ψ and χ, then it also holds for any formula that can be constructed using ψ and/or χ and one connective. Since all of the connectives can be defined in terms of $\neg$, $\wedge$, and $\forall$, it

will be sufficient to show it for $\neg\psi$, $\psi \wedge \chi$, and $\forall\mu\psi$. It is clearly true for $\varphi \wedge \psi$; we will prove it for $\neg\psi$ and $\forall\mu\psi$.

For $\neg\psi$: if ν occurs free in both positively and negatively embedded belief contexts in $\neg\psi$, then the induction hypothesis is preserved trivially, because neither of the antecedents of (a) or (b) holds. Otherwise, assume first that ν does not occur in any positively embedded belief context in $\neg\psi$, which means that it does not occur in any negatively embedded belief context in ψ. If $M_2 \models \neg\psi$, then $M_2 \not\models \psi$. By the induction hypothesis, it follows that $M_1 \not\models \psi_{\tau/\nu}$, and therefore $M_1 \models \neg\psi_{\tau/\nu}$, preserving the hypothesis. A similar argument applies in the opposite direction if ν occurs in no negatively embedded belief context in $\neg\psi$.

For $\forall\mu\psi$: assume for case (a) that $M_1 \models \forall\mu\psi_{\tau/\nu}$. That means that for any model M_1' that differs from M_1 at most in its interpretation of μ, $M_1' \models \psi_{\tau/\nu}$. For each such M_1' there is an M_2' identical to M_2 except that $|\mu|^{M_2'} = |\mu|^{M_1'}$. μ does not occur in τ (otherwise substituting τ for ν would cause the free occurrences of μ in τ to become bound) and is not the same as ν (otherwise ν would not occur free in $\forall\mu\psi$), so $|\nu|^{M_2'}$ is $|\tau|^{M_1'}$ (because $|\nu|^{M_2}$ is $|\tau|^{M_1}$). The induction hypothesis therefore applies, entailing that $M_2' \models \psi$ for all of the M_2' under consideration, so $M_2 \models \forall\mu\psi$ by the semantics of $\forall$. Again, a similar argument applies in the other direction. □

We can now prove the soundness of the inference rule.

Theorem 9 (Soundness of Restricted $\forall$-Instantiation) *For model M, formula φ, variable ν, and term τ such that the substitution of τ for ν in φ would not cause any free variable of τ to become bound, and such that no occurrence of ν that is free in φ is within a belief context that is positively embedded in φ, if $M \models \forall\nu\varphi$ then $M \models \varphi_{\tau/\nu}$.*

Proof: Assume $M \models \forall\nu\varphi$. Then $M' \models \varphi$ for every M' that differs from M only in its interpretation of ν. In particular, this holds for the M' that is identical to M except that $|\nu|^{M'}$ is $|\tau|^{M}$. For that choice of M', we have $M' \models \varphi$, and we have restricted the rule so that no occurrence of ν is in a positively embedded belief context, so by Lemma 8 it follows that $M \models \varphi_{\tau/\nu}$. □

3.3.4 ∃-Elimination and Skolemization

In the completeness proof, we will find it convenient to treat an existential quantifier as notation for a negated universal quantifier, so we formulate the rule of ∃-elimination in those terms:

$$\frac{\Delta \quad \neg\forall\nu\varphi \qquad \begin{array}{c}\Delta, \neg\varphi_{\mu/\nu} \\ \vdots \\ \psi\end{array}}{\psi}$$

where μ and ν are both variables, and μ doesn't occur free in Δ, φ, or ψ.

In words, this rule says that if ψ can be proved from the premises $\Delta \cup \{\neg\varphi_{\mu/\nu}\}$, where μ doesn't occur in Δ, φ, or ψ, then it is also a valid conclusion from the premises $\Delta \cup \{\neg\forall\nu\varphi\}$.

The ∃-elimination rule involves replacing an existentially quantified variable with a free variable. This is similar to Skolemization, in which an existentially quantified variable is replaced with ground term. An ∃-elimination rule that used ground Skolem terms instead of free variables would be sound in FOL; but when applied to a formula in which the existentially quantified variable occurs in a negatively embedded belief context, such a rule would *not* be sound. Consider a belief machine that always answers yes to the query $P(\tau) \vee \neg P(\tau)$ for any ground term τ. For this machine, the sentence $\exists x \neg B(a, [P(x) \vee \neg P(x)])$ is consistent (and satisfiable, namely by models in which some individual is not denoted by any ground term), but no Skolemized version of the sentence is consistent, because the rule of simulative inference can be used to prove $B(a, P(\tau) \vee \neg P(\tau))$ for any ground term τ.

Since our version of ∃-elimination uses free variables instead of ground Skolem terms, it avoids drawing unjustified conclusions. However, if this were the only form of ∃-elimination available, the logic would be weaker than necessary, because using Skolem terms in *positively* embedded belief contexts can lead to sound conclusions that could not be reached otherwise. For example, consider a belief machine that always applies a rule of ∃-introduction, so that if an agent believes $P(\tau)$ for any term τ, then it must also believe $\exists x[P(x)]$. For such a machine, $\exists x[B(a, P(x))]$ entails $B(a, \exists x[P(x)])$, but that conclusion cannot be derived using only the inference rules we have introduced so far. The proof would require an application of the rule of simulative inference, but the belief arguments of the premises of that rule must be sentences, and $P(x)$ is not a sentence. Therefore, we introduce a rule of belief Skolemization:

$$\dfrac{\Delta \quad B(\alpha,\varphi) \qquad\qquad \begin{array}{c}\Delta, B(\alpha,\varphi_{\kappa/\nu}), \kappa = \nu \\ \vdots \\ \psi\end{array}}{\psi}$$

where ν is a free variable of φ, and κ is some "ordinary" individual constant that does not occur in Δ, α, φ, or ψ (the definition of an ordinary constant will be given below).

For the completeness proof, it will be useful to note the following: this rule ensures that if $\{\Delta, B(\alpha,\varphi_{\kappa/\nu}), \kappa = \nu\}$ is inconsistent for some constant κ that doesn't occur in Δ, α, or φ, then $\{\Delta, B(\alpha,\varphi)\}$ is also inconsistent. This is seen by letting ψ in the inference rule be $\perp$.

This rule is not sound for all belief machines. Consider a belief machine that treats the individual constant c differently from all other terms: in some states the machine will answer yes to the query $P(c)$, but no matter what it has been *TELL*ed, it always answers no to $P(\tau)$ for any term τ other than c. For this belief machine, the sentence $\exists x B(a, P(x))$ is satisfiable, but Skolemizing with an arbitrary constant d will result in the unsatisfiable sentence $B(a, P(d))$.

Consider another belief machine that can believe $P(\tau)$ for any constant τ, but only for one such constant in any given belief state. For this machine, each of $B(a, P(b))$ and $B(a, P(c))$ is satisfiable on its own, but their conjunction is not. The theory

$$\{B(a, P(b)), \exists x B(a, P(x))\}$$

is satisfiable, but the theory

$$\{B(a, P(b)), \exists x B(a, P(x)), B(a, P(c))\},$$

which is the result of Skolemizing the original theory with the Skolem constant c, is unsatisfiable.

We can allow belief machines to have constants that receive special treatment, but such constants are not appropriate for use as Skolem constants. As long as there is also an infinite supply of "ordinary" constants, *i.e.* ones about which the belief machine has no *a priori* disposition, belief Skolemization is still possible. The following con-

straint formalizes this requirement; we will then prove that for machines that satisfy constraints C1–C4 and the new constraint, the belief Skolemization rule is sound.

C5 (monotonicity under substitution) *There must be infinitely many individual constants κ such that for any ground term τ and sentences $\varphi_1, \ldots, \varphi_n$,*

1. *if*

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$$

and $\varphi_{1\tau/\kappa}, \ldots, \varphi_{n\tau/\kappa}$ is monotonically acceptable in S_0, then

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_{1\tau/\kappa}, \ldots, \varphi_{n\tau/\kappa}), \psi_{\tau/\kappa}) = yes$$

2. *if*

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots \varphi_n), \varphi_i) = no$$

for some $1 \leq i \leq n$, then

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_{1\tau/\kappa}, \ldots, \varphi_{n\tau/\kappa}), \varphi_{j\tau/\kappa}) = no$$

for some $1 \leq j \leq n$.

The intuition behind this constraint is that if one set of sentences contains no less information than another, then the belief machine should draw no fewer conclusions from the first set than from the second. An assertion about a term that has already been used in other assertions, or about a functional term containing function and/or individual constants that have been used in other assertions, conveys more information than the same assertion about a previously unseen constant (such as a Skolem constant). For example, if we know $P(c)$ but know nothing about d, then the assertion $Q(c)$ conveys more information than $Q(d)$, because it makes a connection to other knowledge. The first half of the constraint says that if the belief machine draws the conclusion ψ after being *TELL*ed something about a Skolem constant κ, then it must draw the corresponding conclusion $\psi_{\tau/\kappa}$ if it is *TELL*ed the same thing about any term τ, unless $\psi_{\tau/\kappa}$ is inconsistent with its previous beliefs. The second half of the constraint says that if a sequence of assertions about a Skolem constant is not acceptable, then the sequence must still not be acceptable if the Skolem constant is replaced by any other term.

The constraint of monotonicity under substitution does allow for the possibility that certain constants have special computational significance to a belief machine. For example, a machine might have hard-coded beliefs about numbers, expressed using the constants $0, 1, 2, \ldots$. For such a machine, the assertion *weight-of*$(a) = 15$ conveys information that is not contained in *weight-of*$(a) = c$, for example, assuming the machine has not seen the constant c before. The constant 15 might not satisfy the constraint of monotonicity under substitution for this machine. If not, then it would of course be inappropriate to use 15 as a Skolem constant, even if the machine had not previously been

*TELL*ed anything involving 15 (*i.e.* if all of its beliefs involving 15 were intrinsic to its mechanism). This is permitted by the constraint, as long as infinitely many non-special constants are available for Skolemization.

We now prove the soundness of proofs that use any of the above inference rules, including $\exists$-elimination and belief Skolemization.

Theorem 10 (Soundness of Proofs) *Given a belief machine m satisfying constraints C1–C5, an m-model M, set of formulas Γ, and formula ψ, if $M \models \Gamma$ and $\Gamma \vdash \psi$ using the inference rules listed above, then $M \models \psi$.*

Proof: The proof is by induction on the length of the derivation $\Gamma \vdash \psi$. For current purposes, we define the length of a proof to include the lengths of any subproofs used in *reductio* and $\exists$-elimination steps. The basis of the induction is proofs of length zero: if $\psi \in \Gamma$, then trivially $M \models \psi$. Assume that every proof of length $\leq n$ is sound. A proof of ψ from Γ of length $n+1$ is composed of a proof of length $\leq n$, followed by a final proof step whose premises are in Γ or were conclusions of earlier steps, and whose conclusion is ψ. If that final step is an application of anything other $\exists$-elimination or belief Skolemization, then the entire proof is sound, because we have already shown that those other rules are sound. We now consider the two remaining rules in turn.

If the final step is an application of $\exists$-elimination, let Δ and $\neg\forall\nu\varphi$ be its premises, and let $\Delta, \neg\varphi[\mu/\nu] \vdash \psi$ be the required subproof. By the induction hypothesis, $M \models \Delta \cup \{\neg\forall\nu\varphi\}$, since the premises of this proof step were all proved in $\leq n$ steps from the original premises Γ. By the semantics of $\neg$ and $\forall$ (*i.e.* the semantics of $\exists$), there is a model M', which differs from M by at most its interpretation of ν, for which $M' \not\models \varphi$, so that $M' \models \neg\varphi$. Construct another model M'' identical to M except that $|\mu|^{M''} = |\nu|^{M'}$. Then $M'' \models \neg\varphi_{\mu/\nu}$. Since μ doesn't occur free in Δ, it is also the case that $M'' \models \Delta$. Since all the premises of the subproof are true in M'', and the subproof is sound by the induction hypothesis, it follows that $M'' \models \psi$. Since M'' and M differ only in their interpretation of μ, which does not occur free in ψ, it also follows that $M \models \psi$.

If the final step is an application of belief Skolemization, let Δ and $B(\alpha, \varphi)$ be its premises, and let $\Delta, B(\alpha, \varphi_{\kappa/\nu}), \kappa = \nu \vdash \psi$ be the required subproof of length $\leq n$. $M \models \Delta \cup \{B(\alpha, \varphi)\}$ by the induction hypothesis, because these premises were derived at earlier steps in the proof. Since $M \models B(\alpha, \varphi)$, there is some variable substitution σ that is extension-preserving under M for which $M \models B(\alpha, \varphi^\sigma)$. Let τ be $\sigma(\nu)$. From the subproof $\Delta, B(\alpha, \varphi_{\kappa/\nu}), \kappa = \nu \vdash \psi$, we can construct another proof of the same length by replacing each occurrence of κ with τ. We must show that the resulting sequence is itself a valid proof, by showing that each step is licensed by one of the inference rules. This can be done case-by-case for each of the inference rules that might be used in the proof. It is clear for all of the rules other than those of positive and negative simulative inference: none of the other rules is ever sensitive to the form of a

ground term. In other words, given a valid application of one of these rules, if one term is uniformly replaced with another in the premises and the conclusion, then the result is still a valid application of the rule. If one of these rules licenses the conclusion ψ from the premises $\varphi_1, \ldots, \varphi_n$, then it also licenses the conclusion $\psi_{\tau/\kappa}$ from the premises $\varphi_{1\tau/\kappa}, \ldots, \varphi_{n\tau/\kappa}$. This is not always the case for the rules of positive and negative simulative inference. These two rules depend on the behavior of the belief machine, which *may* be sensitive to the form of a term (for example, some belief machines may have hard-coded beliefs about the special constant 0 that they do not have about other ground terms). However, the first part of the constraint of monotonicity under substitution (C5) restricts sensitivity to the form of terms in such a way that if simulative inference licenses the conclusion $B(\alpha, \psi)$ from the premises $B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)$, and those premises are simultaneously satisfiable, then simulative inference also licenses the conclusion $B(\alpha, \psi_{\tau/\kappa})$ from the premises $B(\alpha, \varphi_{1\tau/\kappa}), \ldots, B(\alpha, \varphi_{n\tau/\kappa})$. The second part of the constraint ensures that if the rule of negative simulative inference licenses the conclusion $\perp$ from the premises $B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)$, then it also licenses the conclusion $\perp$ from $B(\alpha, \varphi_{1\tau/\kappa}) \ldots B(\alpha, \varphi_{n\tau/\kappa})$.

Since κ was chosen so that it doesn't occur in $\Delta, \alpha, \varphi,$ or ψ, the premises $\Delta_{\tau/\kappa}$ are the same as Δ, and the conclusion $\psi_{\tau/\kappa}$ is the same as ψ. Since τ is $\sigma(\nu)$, and σ is extension-preserving under M, $M \models \tau = \nu$; and since $(\varphi_{\tau/\nu})^\sigma$ is the same as φ^σ, $M \models B(\alpha, (\varphi_{\tau/\nu})^\sigma)$, so $M \models B(\alpha, \varphi_{\tau/\nu})$.

We have constructed a proof $\Delta, B(\alpha, \varphi_{\tau/\nu}), \tau = \nu \vdash \psi$ of length $\leq n$, and we have shown the premises of that proof are true in M. By the induction hypothesis, the proof is sound, so its conclusion ψ must be true in M as well. □

3.4 Completeness

Depending on the choice of the belief machine m, the inference rules given here may or may not be complete, and in fact there are some belief machines for which no complete proof system can exist. However, for a restricted subset of the language, the set of inference rules introduced above is refutation complete for every belief machine.

Konolige [1986] presents a logic of belief that is similar to ours, but for which a general completeness proof does exist. Section 5.5.3 points out an unrealistic assumption that weakens the notion of entailment in Konolige's model, making the completeness proof possible.

3.4.1 General Incompleteness

The proofs in this section are due to Len Schubert.

Consider a very simple belief machine which performs no inference at all. It keeps a list of the sentences it has been *TELL*ed, and says yes to exactly those sentences and no to all others. This machine meets all the criteria for the soundness of our inference rules. Using such a machine, the sentence $\forall x B(a, P(x))$ has only finite models: for this sentence to be true in an infinite model, $B(a, P(\tau))$ would have to be true for infinitely many terms τ, which is impossible since the belief machine can have been *TELL*ed only finitely many sentences.

Let Δ be a theory that has only infinite models. For instance, Δ could be this axiomatization of the "greater-than" relation:

$$\forall x \exists y (y > x)$$
$$\forall x \forall y \forall z (x > y) \wedge (y > z) \supset (x > z)$$
$$\forall x \forall y (x > y) \supset \neg (y > x)$$

Let Δ' be the theory

$$\Delta \cup \{\forall x B(a, P(x))\}.$$

Since Δ has only infinite models, and $\forall x B(a, P(x))$ has only finite models, Δ' is unsatisfiable. However, intuition about the inference rules we have introduced says that Δ' is consistent (has no disproof). If that is so, then the logic is incomplete. A more formal proof will use the following lemma:

Lemma 11 *It is undecidable in general whether a given finite theory of first-order logic has a finite model.*

Proof: If there were a decision procedure for this problem, it could be used to construct a decision procedure for the halting problem. To decide if a given Turing machine halts on a given input, encode the (finite) initial state of the machine and its tape, and a description of how the finite state machine operates, as a finite theory of FOL. This can be done in such a way that any model of the theory will include one individual for each of the finitely many states of the machine, one for each of the finitely many symbols in the tape alphabet, one for each of the spaces on the tape used in the computation, and one for each unit of time at which the machine changes state, moves its tape head, or writes a symbol on the tape. If the machine halts at some time, the theory will have a finite model; if it runs forever, the theory will have only infinite models. □

Theorem 12 *For a belief machine that answers affirmatively to those sentences it has explicitly been TELLed, and no others, the logic is incomplete.*

Proof: If there were a complete proof system for our logic, then it could be used to decide whether an arbitrary theory Δ has a finite model, contrary to Lemma 11. If Δ had no finite model, then

$$\Delta' = \Delta \cup \{\forall x B(a, P(x))\}$$

would be unsatisfiable, and given a complete proof system we could eventually prove this. If Δ did have a finite model, then one could eventually be found simply by picking an arbitrary domain of cardinality n for each $n = 1, 2, \ldots$, each time enumerating and testing all possible models (for the subset of L built only from vocabulary occurring in Δ') with that domain. $\Box$

This proof also shows that given certain belief machines, our logic lacks the property of compactness: there can be an infinite unsatisfiable theory that has no unsatisfiable finite subset.

3.4.2 Restricted Completeness

We have shown that for some belief machines, no set of inference rules is complete. However, we will now show that the set of rules introduced above is complete for a restricted subset of the language, given any belief machine for which the rules are sound. Loosely stated, the restriction is that universal quantification into a positively embedded belief context is not allowed.

As mentioned in Section 3.3.3, to make the definition of a positive embedding precise, we treat $\exists\nu\psi$ as an abbreviation for $\neg\forall\nu\neg\psi$, and $\psi \supset \chi$ as an abbreviation for $\neg\psi \vee \chi$. Another complication is that universal quantification can be "disguised" as existential quantification by nesting an existential formula inside the scope of a universal quantifier: $\forall x \exists y[x = y \wedge B(a, P(y))]$ is logically equivalent to $\forall x B(a, P(x))$. For the completeness theorem, we define L_1 to be a language constructed like L, but with the restriction that when all existential quantifiers are rewritten as universal quantifiers with negation, no free variable of the belief argument of a positively embedded belief atom is bound by a positively embedded (universal) quantifier, nor by a negatively embedded (universal) quantifier that is inside the scope of a positively embedded one.

Theorem 13 (Restricted Refutation Completeness) *If formula $\varphi \in L_1$ is unsatisfiable, then $\varphi \vdash \perp$.*

Proof: The proof is an adaptation of the completeness proof for standard FOPC found in [Hodges, 1983]. There are two main steps: first, it is shown that any consistent formula $\varphi \in L_1$ can be extended to a Hintikka set $\Delta_\infty^{Sk=}$ in which only finitely many formulas are asserted to be beliefs of each agent; and second, that every such set has a model. Any model for $\Delta_\infty^{Sk=}$ is a model for φ (since $\varphi \in \Delta_\infty^{Sk=}$), and therefore every consistent formula has a model. This is the contrapositive of the refutation completeness theorem, so the theorem itself follows.

To begin, choose an ordering $\psi_0, \psi_1, \psi_2, \ldots$ of the formulas of L_1 such that every formula occurs infinitely often in the list; and choose some ordering $\tau_0, \tau_1, \tau_2, \ldots$ of all the terms in L_1 so that every term in the language has a finite index.

To extend an arbitrary formula $\varphi \in L_1$ to a Hintikka set of formulas in L_1, we define theories $\Delta_0, \Delta_1, \ldots$ by induction. Δ_0 is the set $\{\varphi\}$. For $i \geq 0$, every formula in Δ_i is in Δ_{i+1}; in addition, if the formula ψ_i (from the ordering chosen above) is an element of Δ_i, then certain new formulas are elements of Δ_{i+1}, depending on the form of ψ_i:

1. if ψ_i is of the form $\neg\neg\chi$, then $\chi \in \Delta_{i+1}$.
2. if ψ_i is of the form $\neg(\chi \wedge \omega)$, then $\neg\chi \vee \neg\omega \in \Delta_{i+1}$.
3. if ψ_i is of the form $\neg(\chi \vee \omega)$, then $\neg\chi \wedge \neg\omega \in \Delta_{i+1}$.
4. if ψ_i is of the form $\chi \wedge \omega$, then both $\chi, \omega \in \Delta_{i+1}$.
5. if ψ_i is of the form $\chi \vee \omega$, then either χ or ω is in Δ_{i+1}. The choice is made such that Δ_{i+1} is consistent.
6. if ψ_i is of the form $\neg\forall\nu\chi$, and there is no variable μ for which $\neg\chi_{\mu/\nu}$ is already in Δ_i, then $\neg\chi_{\mu/\nu}$ is in Δ_{i+1} for some new variable μ that doesn't occur in Δ_i.
7. if ψ_i is of the form $\forall\nu\chi$, then $\chi_{\tau_j/\nu}$ is in Δ_{i+1} for the first term τ_j from the ordering chosen above for which $\chi_{\tau_j/\nu}$ isn't already in Δ_i and for which the substitution doesn't cause any free variable of τ_j to become bound.

We need to prove that at each step, if Δ_i is consistent, then there exists a Δ_{i+1}, as described in the appropriate one of the above cases, which is also consistent. It is immediately clear that no inconsistency will be introduced by cases 1, 2, 3, or 4, because the new formulas they introduce correspond to inferences licensed by the rule of double negation, DeMorgan's rules for $\wedge$ and $\vee$, and $\wedge$-elimination, respectively. Since the new formulas were derivable from the formulas in Δ_i, and Δ_i is consistent, the new formulas must be consistent with Δ_i. The new formula $\neg\chi_{\mu/\nu}$ introduced in case 6 is not licensed as a sound inference, but if $\Delta_i \cup \{\neg\chi_{\mu/\nu}\} \vdash \bot$, then $\Delta_i \vdash \bot$ by the rule of $\exists$-elimination.

Case 5 is not as simple, because it involves making a choice. Either χ or ω, but not necessarily both, is added to Δ_{i+1}. But one of the two choices must result in a consistent Δ_{i+1}. If not, *i.e.* if there were proofs $\Delta_i, \chi \vdash \bot$ and $\Delta_i, \omega \vdash \bot$, then there would also be *reductio* proofs $\Delta_i \vdash \neg\chi$ and $\Delta_i \vdash \neg\omega$. From there, one could construct a proof $\Delta_i \vdash \neg(\chi \vee \omega)$ using $\wedge$-introduction and DeMorgan's rule. Since $\chi \vee \omega \in \Delta_i$, that would mean that Δ_i was inconsistent, contrary to assumption.

Case 7 corresponds to the the rule of $\forall$-instantiation. For that rule to be applicable, the variable ν must not occur in a positively embedded belief context. This restriction is met if the formula ψ is in L_1; and indeed it must be, because we assumed that φ is in L_1, and none of the cases 1–7 introduces any new quantifiers, nor changes the polarity of the embedding of any quantifier.

We have shown that each Δ_i exists and is consistent. Let Δ_∞ be the union of all the Δ_i. For Δ_∞ to be inconsistent would mean that there was a proof of $\perp$ from some subset of it. Since proofs are finite by definition, that subset would have to be finite. Any finite subset of Δ_∞ is contained in Δ_i for some finite i, so if Δ_∞ were inconsistent then some Δ_i would have to be inconsistent. Therefore, Δ_∞ exists and is consistent.

Next we will show that, while there may be infinitely many belief atoms that are elements of Δ_∞, only finitely many formulas occur as the belief arguments of such belief atoms. Cases 1–7 break down complex formulas into their component subformulas. When a new formula is introduced into some set Δ_i, where $i \geq 1$, it is always because of the presence of some other formula in Δ_{i-1}, and each atom occurrence in the new formula corresponds to one atom occurrence in the old one. Therefore, the ancestry of each atom occurrence in Δ_∞ can be traced back to one atom occurrence in φ. We begin by showing that if two belief atoms are themselves elements of Δ_∞ (as opposed to subformulas of elements of Δ_∞) and have the same φ-ancestor, then they have identical belief arguments. In cases 1–6, each atom occurrence in the chosen ψ_i is given at most one immediate descendant. In other words, the ancestry tree only branches in case 7. Since every Δ_i is a subset of L_1, if ψ_i is a universally quantified formula, then there is no quantification (neither existential nor universal) into positively embedded belief contexts in ψ_i. Therefore, all descendants of positively embedded belief atom occurrences in ψ_i have identical belief arguments (since the only cases in which a child is not identical to its parent are those involving quantification). The ancestry tree rooted at a given positively embedded belief atom occurrence in φ is linear up to the first application of case 7, and after that point all descendants have the same belief argument, so it follows that all unembedded belief atoms that are elements of Δ_∞ and descendants of the same φ-ancestor have the same belief argument.

Since φ is a finite formula, it contains only finitely many belief atom occurrences, and, in particular, only finitely many positively embedded ones. We have shown that all descendants of each such occurrence that are (unembedded) elements of Δ_∞ have identical belief arguments. Therefore, only finitely many formulas occur as belief arguments in belief atoms that are elements of Δ_∞.

We now construct another theory, Δ_∞^{Sk}, which contains all of the formulas in Δ_∞, plus Skolemized versions of any top-level positive belief literals whose belief arguments have free variables. The terms that occur in Δ_∞ can be partitioned into equivalence classes such that terms τ_1 and τ_n are in the same class iff there is a sequence $\tau_1, \ldots, \tau_n$ such that the equations $\tau_1 = \tau_2, \tau_2 = \tau_3, \ldots, \tau_{n-1} = \tau_n$ are all elements of Δ_∞ (treating the order of an equation as unimportant, so that an equation $\tau_2 = \tau_1$ is just as good as $\tau_2 = \tau_1$). Let $\Pi_1, \Pi_2, \ldots$ be all the equivalence classes of terms that occur in Δ_∞, and let $\tau_{i,1}, \tau_{i,2}, \ldots$ be all the terms in equivalence class Π_i. For each equivalence class Π_i, there is a set of formulas

$$\{\psi | B(\tau_{i,j}, \psi) \in \Delta_\infty \text{ for some j}\}.$$

We have shown that there are only finitely many of these formulas; call them $\psi_{i,1}, \ldots, \psi_{i,n_i}$. For each formula $\psi_{i,j}$, let $\psi'_{i,j}$ be a Skolemized version, in which all of the free variables $\nu_{i,j,1}, \ldots, \nu_{i,j,m}$ are replaced by previously unused Skolem constants $\kappa_{i,j,1}, \ldots, \kappa_{i,j,m}$, respectively. To form Δ^{Sk}_{∞}, we add to Δ_{∞} the formulas $B(\tau_{i,1}, \psi'_{i,1}), \ldots, B(\tau_{i,1}, \psi'_{i,n_i})$ ($\tau_{i,1}$ is an arbitrarily chosen member of the equivalence class Π_i) and the equations $\nu_{i,j,k} = \kappa_{i,j,k}$. Since the number of $\psi_{i,j}$ for each equivalence class Π_i was finite, and we add only one $\psi'_{i,j}$ for each $\psi_{i,j}$, the number of formulas that occur as belief arguments of each equivalence class is still finite.

We now show that Δ^{Sk}_{∞} must be consistent: for each formula $B(\tau_{i,1}, \psi'_{i,j})$ in $\Delta^{Sk}_{\infty} - \Delta_{\infty}$, the un-Skolemized version $B(\tau_{i,1}, \psi_{i,j})$ can be derived from Δ_{∞} by the rule of substitution of equal terms. This is because $B(\tau_{i,k}, \psi_{i,j})$ is in Δ_{∞} for some k, and $\tau_{i,1}$ and $\tau_{i,k}$ are in the same equivalence class. $\psi'_{i,j}$ is $\psi_{i,j}$ with free variables replaced by Skolem constants, and an equation between each free variable and its respective Skolem constant is also in Δ^{Sk}_{∞}. Therefore, if there were a proof of $\perp$ from premises in Δ^{Sk}_{∞}, then there would also be a proof of $\perp$ from Δ_{∞} alone, using the rule of belief Skolemization. We have already shown that Δ_{∞} is consistent, so Δ^{Sk}_{∞} must be as well.

Finally, we construct another theory $\Delta^{Sk=}_{\infty}$, which is the closure of Δ^{Sk}_{∞} under the application of the rule of substitution of equals; that is, if Δ^{Sk}_{∞} contains an equation $\alpha = \beta$ and a formula φ, then any formula φ' that can be constructed by replacing one or more instances of α that are outside of belief contexts with β is in $\Delta^{Sk=}_{\infty}$, provided that none of the occurrences of α being replaced contain variables bound in φ, and such that the substitution doesn't cause any free variable of β to become bound in φ'. While this may introduce new equations (whenever the formula φ is itself an equation), it does not merge any two equivalence classes from Δ_{∞}. Each equivalence class in $\Delta^{Sk=}_{\infty}$ is an equivalence class from Δ_{∞} with the addition of some Skolem constants. Also, since the substitution of one term for another does not apply within belief contexts, there are still only finitely many formulas that occur as belief arguments of each equivalence class. $\Delta^{Sk=}_{\infty}$ must be consistent, because Δ^{Sk}_{∞} was consistent, and $\Delta^{Sk=}_{\infty}$ is simply the closure of Δ^{Sk}_{∞} under one of the inference rules.

Having shown how to extend a given formula φ to a Hintikka set $\Delta^{Sk=}_{\infty}$ in which only finitely many belief sentences are associated with each equivalence class of agent terms, we will now show that any such set has a model. Given an arbitrary belief machine $m = \langle \Gamma, S_0, \textit{TELL}, \textit{ASK} \rangle$, we construct an m-model $M = \langle D, I, \gamma \rangle$ that satisfies all formulas in $\Delta^{Sk=}_{\infty}$, including φ itself. The domain D is the set $\{\Pi_1, \Pi_2, \ldots\}$ of equivalence classes of terms that occur in $\Delta^{Sk=}_{\infty}$. The interpretation function I maps each individual constant and free variable to the equivalence class of which it is a member; it maps each n-ary predicate constant π to the set of all tuples of equivalence classes $\langle \Pi_1, \Pi_2, \ldots, \Pi_n \rangle$ such that a formula $\pi(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \Delta^{Sk=}_{\infty}$, where each $\alpha_i \in \Pi_i$; and it maps each n-ary function constant θ to a function from a sequence of n equivalence classes to another equivalence class, such that $I(\theta)(\Pi_{i_1}, \ldots, \Pi_{i_n})$ is the equivalence class of the term $\theta(\kappa_{i_1}, \ldots, \kappa_{i_n})$, where each κ_{i_j} is any member of Π_{i_j}.

For each equivalence class $\Pi_i \in D$, we have shown that only finitely many sentences $\psi'_{i,1}, \ldots, \psi'_{i,n_i}$ occur as the second argument of a belief atom that is an element of $\Delta_\infty^{Sk=}$ whose first argument is in Π_i. We define the function γ, which maps each agent to its belief state, so that

$$\gamma(\Pi_i) = \textit{TELL}(S_0, \psi'_{i,1}, \psi'_{i,2}, \ldots, \psi'_{i,n_i}).$$

Now that we have defined the model M, it remains to be shown that all of the formulas in $\Delta_\infty^{Sk=}$, including φ, are true in M. This is done by induction on the complexity of formulas. For present purposes, we define the complexity of a formula to be the number of occurrences of $\forall, \wedge$, and $\vee$ in the formula, not counting those inside belief atoms. We continue to consider $\exists$ and $\supset$ as defined in terms of the other operators, and we do not count occurrences of $\neg$ in the complexity because we handle negation at each step of the induction.

As the base case, we need to show that for any atomic formula χ, if χ preceded by an even number of negations, or χ itself, is an element of $\Delta_\infty^{Sk=}$, then $M \models \chi$, and if χ preceded by an odd number of negations is an element of $\Delta_\infty^{Sk=}$ then $M \not\models \chi$. Case 1 ensures that if χ with an even number of negations is in $\Delta_\infty^{Sk=}$ then χ itself is, and if χ with an odd number of negations is in $\Delta_\infty^{Sk=}$ then $\neg\chi$ is. Therefore, it suffices (for the base case) to show that the atomic elements of $\Delta_\infty^{Sk=}$ are true, and that atoms whose negations are elements of $\Delta_\infty^{Sk=}$ are not true. This is clearly the case for ordinary (non-belief) atoms, by the way we constructed the interpretation I (the argument is the same as in the proof for FOL), so we will prove it only for belief atoms.

For the case of top-level positive belief literals, note that all of the sentences $\psi'_{i,1}, \ldots, \psi'_{i,n_i}$ that we *TELL*ed to the ith agent's belief machine must be believed in the resulting state. If they were not, ***i.e.*** if

$$\textit{ASK}(\textit{TELL}(S_0, \psi'_{i,1}, \ldots, \psi'_{i,n_i}), \psi'_{i,j}) = no$$

for some $1 \leq j \leq n_i$, then $\Delta_\infty^{Sk=}$ would have been inconsistent: the rule of negative simulative inference would license the conclusion $\perp$ from the premises $B(\alpha, \psi'_{i,1}), \ldots, B(\alpha, \psi'_{i,n_i})$ for any α in equivalence class Π_i. Since $\psi'_{i,1}, \ldots, \psi'_{i,n_i}$ are all believed in the belief state that M assigns to agent Π_i, M satisfies all of the positive, top-level belief literals in $\Delta_\infty^{Sk=}$ whose belief arguments are closed, namely $B(\alpha, \psi'_{i,j})$ for all i, j, and α such that α is in the equivalence class Π_i. Furthermore, for any top-level positive belief literal $B(\alpha, \omega) \in \Delta_\infty^{Sk=}$ whose belief argument ω has free variables $\nu_1, \ldots, \nu_n$, the step that constructed $\Delta_\infty^{Sk=}$ from Δ_∞ resulted in the introduction of a Skolemized form $B(\alpha, \omega_{\kappa_1/\nu_1, \ldots, \kappa_n/\nu_n})$ and the equalities $\kappa_1 = \nu_1, \ldots, \kappa_n = \nu_n$ into $\Delta_\infty^{Sk=}$ as well. A variable substitution σ that maps each ν_i to the corresponding κ_i is extension-preserving under M, and $M \models B(\alpha, \omega^\sigma)$, so $M \models B(\alpha, \omega)$.

The remaining part of the base case is the top-level negative belief literals. Consider a literal $\neg B(\alpha, \omega) \in \Delta_\infty^{Sk=}$. If $M \not\models \neg B(\alpha, \omega)$, then $M \models B(\alpha, \omega)$, so there is some

variable substitution σ that is extension-preserving under M for which $M \models B(\alpha, \omega^\sigma)$. Where Π_i is the equivalence class of term α, this means that

$$\mathit{ASK}(\mathit{TELL}(S_0, \psi'_{i,1}, \ldots, \psi'_{i,n_i}), \omega^\sigma) = yes.$$

Then the rule of simulative inference licenses the conclusion $B(\alpha, \omega^\sigma)$ from $\Delta_\infty^{Sk=}$. Let $\nu_1, \ldots, \nu_m$ be all of the free variables of ω. For σ to be extension-preserving under the constructed model, the equations $\sigma(\nu_1) = \nu_1, \ldots, \sigma(\nu_m) = \nu_m$ must be elements of $\Delta_\infty^{Sk=}$. But then m applications of the rule of substitution in negative belief contexts to the literal $\neg B(\alpha, \omega)$ would license the conclusion $\neg B(\alpha, \omega^\sigma)$, so $\Delta_\infty^{Sk=}$ would be inconsistent, which it is not.

The induction step is to show that if every element of $\Delta_\infty^{Sk=}$ of complexity at most i is true in M, then the same holds for those elements of $\Delta_\infty^{Sk=}$ of complexity $i+1$. Once again, case 1 of the construction makes it only necessary to prove this for formulas with zero or one top-level negation. For a non-negated element of $\Delta_\infty^{Sk=}$ whose complexity is $i+1$ and whose outermost operator is $\wedge$, case 4 ensures that each of the conjuncts is also an element of $\Delta_\infty^{Sk=}$. Since each of them is of complexity $\leq i$, the induction hypothesis says that each of them is true in M and therefore the conjunction is also true. Similarly, if the outermost operator is $\vee$, case 5 ensures that at least one of the disjuncts is in $\Delta_\infty^{Sk=}$, and therefore true in M, so that the whole disjunction is true. If the formula is of the form $\forall\nu\omega$, then case 7 ensures that $\omega_{\tau/\nu} \in \Delta_\infty^{Sk=}$ for every term τ. All of these $\omega_{\tau/\nu}$ are of complexity i, so by the induction hypothesis they are all true in M. The terms of the language cover all of the individuals of M, so the universal formula $\forall\nu\omega$ is also true in M.

We are now left with the negated formulas of complexity $i+1$. Cases 2 and 3 distribute negation over conjunction and disjunction, so the proofs for negated conjunctions and disjunctions are reduced to those for non-negated disjunctions and conjunctions, respectively. For each negated, universally quantified formula $\neg\forall\nu\omega \in \Delta_\infty^{Sk=}$, case 6 ensures that there is a term τ for which $\neg\omega_{\tau/\nu} \in \Delta_\infty^{Sk=}$. Since this latter formula is of complexity i, the induction hypothesis says that it is true in M, and consequently so is the negated universal formula. □

3.5 Some Common Axioms

We now examine some of the properties that the belief relation has in other logics of belief, and discuss the kind of computation the belief machine must perform in order for these properties to hold in our model.

The classical modal logics (see [Hughes and Cresswell, 1968] for example) are characterized by various combinations of a rule of inference and five axioms:

N. If $\vdash \varphi$, conclude $B(\alpha, \varphi)$
K. $B(\alpha, \varphi \supset \psi) \supset (B(\alpha, \varphi) \supset B(\alpha, \psi))$
T. $B(\alpha, \varphi) \supset \varphi$
D. $\neg B(\alpha, \varphi \wedge \neg\varphi)$
4. $B(\alpha, \varphi) \supset B(\alpha, B(\alpha, \varphi))$
5. $\neg B(\alpha, \varphi) \supset B(\alpha, \neg B(\alpha, \varphi))$

The classical modal logics are all built around the idealization that agents are logically omniscient. This condition is obtained by the inclusion of **N** and **K** in all of the traditional axiomatizations. Our computational model of belief was designed explicitly to avoid logical omniscience, and therefore there is no belief machine for which any of the traditional axiomatic bases is valid. However, some of the axioms are interesting in their own right, and are by themselves valid for certain belief machines.

N: Rule **N** is the rule of epistemic necessitation. It says that agents believe all theorems (sentences derivable from just the logical axioms). Since our logic is an extension of first-order logic, it is undecidable in general whether a given sentence is a theorem. Therefore, while there are machines for which this rule is sound (for example, the machine that always answers yes to everything), none of them implement reasonable inference techniques.

K: Axiom **K**, also known as the distribution axiom, is valid for belief machines for which

$$\text{if } \mathit{ASK}(S, \varphi) = yes \text{ and } \mathit{ASK}(S, \varphi \supset \psi) = yes \text{ then } \mathit{ASK}(S, \psi) = yes.$$

Although Levesque's influential approach to the elimination of logical omniscience involved eliminating this axiom, it alone does not ensure logical omniscience. Belief machines for which the axiom is valid are easily constructed.

T: Axiom **T** says that everything an agent believes (or, more commonly, knows) is true. This axiom is usually taken to be the one that differentiates between knowledge and belief, the difference being that one can have false beliefs, but not false knowledge. Our model is a model of belief, not knowledge; the only belief machines for which axiom **T** is valid are those that answer yes only to tautologies, no matter what state they are in.

Axiom **T** can be thought of as describing the way an agent uses its belief machine, as well as describing the belief machine itself. If an agent only *TELL*s its belief machine sentences that are true, and the machine only makes sound inferences, then axiom **T** holds.

D: Axiom **D** is sometimes proposed as an alternative to **T** for describing belief instead of knowledge. It says that no agent can believe a contradiction. In the classical logics, belief sets are closed under logical consequence, so any inconsistent belief set must contain an explicit contradiction. However, axiom **D** by itself only prohibits belief sets that contain an explicit contradiction, so there do exist belief machines for which **D** (but not both **N** and **K**) is valid. These are machines for which

$$\mathit{ASK}(S, \varphi \wedge \neg\varphi) = no \tag{3.14}$$

for every belief state S and sentence φ. It is a simple matter for a machine to satisfy this constraint, by performing a syntactic check on all queries to see if they are of the form $\varphi \wedge \neg\varphi$ for any φ, and if so answering no. However, it is less obvious that there are machines that satisfy this constraint and are also reasonably competent at handling conjunctions. For example, for a machine to satisfy both (3.14) and the constraint

$$\mathit{ASK}(S, \varphi \wedge \psi) = yes \text{ iff } \mathit{ASK}(S, \varphi) = \mathit{ASK}(S, \psi) = yes$$

requires more involved computation.

There are in fact machines that satisfy (3.14) and also reason competently about the Boolean connectives: consider a machine whose *ASK* function is implemented by another function *DECIDE*, which is similar to *ASK* but has three possible values instead of two. Let those values be $1, 1/2,$ and 0, meaning true, unknown, and false, respectively. *DECIDE* could be a recursive function defined as follows:

$$\begin{aligned}
&\mathit{DECIDE}(S, \neg\varphi) = 1 - \mathit{DECIDE}(S, \varphi)\\
&\mathit{DECIDE}(S, \varphi \wedge \psi) = \min(\mathit{DECIDE}(S, \varphi), \mathit{DECIDE}(S, \psi))\\
&\mathit{DECIDE}(S, \varphi \vee \psi) = \max(\mathit{DECIDE}(S, \varphi), \mathit{DECIDE}(S, \psi))\\
&\mathit{DECIDE}(S, \varphi \supset \psi) = \max(1 - \mathit{DECIDE}(S, \varphi), \mathit{DECIDE}(S, \psi))
\end{aligned}$$

If *ASK* is defined as

$$\mathit{ASK}(S, \varphi) = \begin{cases} yes \text{ if } \mathit{DECIDE}(S, \varphi) = 1 \\ no \text{ otherwise} \end{cases}$$

then (3.14) is clearly satisfied, and the machine is a complete propositional reasoner.

4 and 5 (Introspection): Axiom **4** is the positive introspection axiom. It says that if an agent believes a sentence φ, then it believes that it believes φ. Using the indexical constant me introduced in Section 2.4, one can express the positive introspection axiom as

$$B(\alpha, \varphi) \supset B(\alpha, B(me, \varphi)).$$

The axiom is valid for all machines for which

$$\text{if } \mathit{ASK}(S, \varphi) = yes \text{ then } \mathit{ASK}(S, B(me, \varphi)) = yes.$$

There are non-trivial machines that satisfy the positive introspection constraint as well as constraints C1–C5. For a simple example, consider a machine that answers a query affirmatively if and only if it can prove the query from the premises on a list of previously *TELL*ed sentences by applying the rule of conjunction splitting and DeMorgan's rules. This machine satisfies C1–C5, and still does so if it is modified so that it makes self-queries to prove sentences of the form $B(me, \varphi)$.

The converse of the positive introspection axiom,

$$B(\alpha, B(me, \varphi)) \supset B(\alpha, \varphi),$$

is also known as the positive faithfulness axiom. It is valid under the converse constraint,

$$\text{if } \textit{ASK}(S, B(me, \varphi)) = yes \text{ then } \textit{ASK}(S, \varphi) = yes.$$

The example machine could also be made to satisfy this constraint: when *TELL*ed a new sentence $B(me, \varphi)$, it could *TELL* itself φ as well.

The negative introspection axiom can be written as

$$\neg B(\alpha, \varphi) \supset B(\alpha, \neg B(me, \varphi)).$$

This axiom is made valid by the semantic constraint

$$\text{if } \textit{ASK}(S, \varphi) = no \text{ then } \textit{ASK}(S, \neg B(me, \varphi)) = yes.$$

It is simple to build a machine that satisfies this constraint, using a similar construction to the one used for positive introspection. When queried about a sentence $\neg B(me, \varphi)$, the machine could query itself about the sentence φ, and answer yes if the answer to the sub-query was no. However, unlike for the positive introspection constraint, only machines whose introspection is trivial can satisfy both this constraint and the monotonicity constraint. For a non-trivial machine to satisfy the monotonicity constraint, every sentence believed in the initial state must be believed in all states, because otherwise no sequence of *TELL*s would be monotonically acceptable. Therefore, if a machine believes $\neg B(me, \varphi)$ in the initial state, and it satisfies the monotonicity constraint, then it must believe $\neg B(me, \varphi)$ in all states, including ones in which it believes φ.

Since a machine with non-trivial negative introspection must violate the monotonicity constraint, our original positive simulative inference rule is not sound for such a machine. However, monotonicity is not required for the soundness of negative simulative inference, and as we will show in Section 3.6, when combined with a rule of negative introspection the negative simulative inference rule can in a sense take over the role of the positive rule.

Negative faithfulness, the converse of negative introspection, is characterized by the axiom

$$B(\alpha, \neg B(me, \varphi)) \supset \neg B(\alpha, \varphi),$$

which is valid for machines satisfying the constraint

$$\text{if } \textit{ASK}(S, \neg B(me, \varphi)) = yes \text{ then } \textit{ASK}(S, \varphi) = no.$$

The example machine would satisfy this constraint if it simply never changed state in response to a *TELL* of the form $\neg B(me, \varphi)$.

Quantifier Raising and Lowering: The Barcan formula,

$$(\forall \nu B(\alpha, \varphi)) \supset B(\alpha, \forall \nu \varphi)$$

is valid only for trivial belief machines that answer yes to every query of the form $\forall \nu \varphi$. The constraint

$$\begin{array}{l}\text{if } \textit{ASK}(S, \varphi_{\tau/\nu}) = yes \text{ for every term } \tau, \\ \text{then } \textit{ASK}(S, \forall \nu \varphi) = yes \text{ for any variable } \nu\end{array}$$

describes belief machines that perform universal generalization, but does not ensure the validity of the Barcan formula because of the possibility that $\forall \nu B(\alpha, \varphi)$ might be true "by accident:" if α believes $\varphi_{\tau/\nu}$ individually for each term τ in some set, and that set happens to cover the entire domain, then $\forall \nu B(\alpha, \varphi)$ is true, even if there are many other terms τ for which α does *not* believe $\varphi_{\tau/\nu}$.

However, it is worth noting that certain assumptions about *ASK* and *TELL* can render the Barcan formula true in all sufficiently large models (and thus in all infinite models). In particular, suppose that we make the following assumptions.

1. *ASK* answers the query $\varphi_{\tau/\nu}$, whenever τ is a ground term that has not appeared in its *TELL*-history, by (a) checking if it has seen at least n constants in its *TELL*-history (say for some fixed n like $n = 10$) that it believes to be distinct (by getting yes when it *ASK*s itself $\neg(\tau = \tau')$), and if not, answering no; (b) otherwise, *ASK*ing itself $\varphi_{\tau'/\nu}$ for every ground term τ' it has seen in its *TELL*-history, and if all answers are yes, answering yes to $\varphi_{\tau/\nu}$, else no;

2. *ASK* answers $\forall \nu \varphi$ by *ASK*ing itself $\varphi_{\kappa/\nu}$ for some constant κ that has not appeared in its *TELL*-history, and answering yes if the answer to $\varphi_{\kappa/\nu}$ is yes, and answering no otherwise

In effect, such a machine would make inductive generalizations based on some number of instances that it believes to be distinct. Of course, it may be mistaken about the distinctness beliefs. But it's reasonable to suppose that agents are connected to the world, *e.g.* through perception, in such a way that they are not too likely to be wrong about distinctness. The inductive generalization may still be incorrect even if the distinctness beliefs are correct, but that is an unavoidable risk.[2] Now observe that if $\forall \nu B(\alpha, \varphi)$

[2]This is not to say that (1a) couldn't be improved to reflect a more subtle theory of induction.

holds and the domain of individuals is larger than the number of ground terms that have appeared in the agent's *TELL*-inputs, then *ASK*$(S, \varphi_{\tau/\nu})$ must be yes for some ground term τ that has not appeared among the *TELL*-inputs (where S is the state of α's belief machine), and hence by (1b) *ASK*$(S, \varphi_{\tau'/\nu}) = yes$ for all ground terms τ', and so by (2) *ASK*$(S, \forall\nu\varphi) = yes$ as well, verifying the Barcan formula.

The converse Barcan formula,

$$B(\alpha, \forall\nu\varphi) \supset \forall\nu B(\alpha, \varphi)$$

is a theorem of the extension of modal system **T** to predicate logic. It is valid only for trivial belief machines that answer no to every sentence of the form $\forall\nu\varphi$. A sentence of the form $\forall\nu B(\alpha, \varphi)$, where ν occurs in φ, is satisfied in a model if for each individual in the domain, there is some term τ denoting that individual for which α believes the sentence $\varphi_{\tau/\nu}$. If $B(\alpha, \forall\nu\varphi)$ is satisfied by some model M, then it is also satisfied by a model identical to M except that its domain contains one extra individual which is not denoted by any term. Such a model does not satisfy $\forall\nu B(\alpha, \varphi)$.

The constraint

> if *ASK*$(S, \forall\nu\varphi) = yes$ for some variable ν
> then *ASK*$(S, \varphi_{\tau/\nu}) = yes$ for any term τ,

is satisfied by machines that do $\forall$-elimination, and imposes a condition similar to the converse Barcan formula, but with an exception allowed for individuals that are not denoted by any term.

Instances of the schema

$$\exists\nu B(\alpha, \varphi) \supset B(\alpha, \exists\nu\varphi)$$

are also theorems of the extension of **T** to predicate logic. The schema is valid for machines that satisfy the constraint

> if there is a term τ for which *ASK*$(S, \varphi_{\tau/\nu}) = yes$,
> then *ASK*$(S, \exists\nu\varphi) = yes$

i.e. machines that perform $\exists$-introduction.

We have considered a number of classical axioms about belief, and seen that only some of them can be realized by belief machines. Those that can't be realized, or can only be realized trivially, describe idealizations that ascribe computationally unrealistic properties to believers. The ones that can be realized non-trivially describe machines with interesting and perhaps desirable inferential capabilities.

3.6 The Simulative Inference Rule for Introspective Machines

We showed in Section 3.5 that the original simulative inference rule is not sound for belief machines with negative introspection, because they violate the monotonicity constraint. However, the negative simulative inference rule does not require that the monotonicity constraint hold, and as we will now show, the combination of the negative simulative inference rule and a rule of negative introspection ($\frac{\neg B(\alpha,\varphi)}{B(\alpha,\neg B(me,\varphi))}$, or equivalently the axiom of negative introspection plus *modus ponens*) is complete in the following sense: for any finite set Φ of belief literals whose belief arguments are closed formulas, if Φ is unsatisfiable, then there is a refutation proof $\Phi \vdash \perp$.

Theorem 14 (Completeness for Belief Literals) *For any belief machine satisfying the closure, acceptable basis, and commutativity constraints, and the negative introspection and negative faithfulness constraints, and for any finite set of belief literals*

$$\Phi = \{B(\alpha,\varphi_1),\ldots,B(\alpha,\varphi_n),\neg B(\alpha,\psi_1,),\ldots,\neg B(\alpha,\psi_m)\},$$

if Φ is unsatisfiable then $\Phi \vdash \perp$ using the rules of negative simulative inference and negative introspection.

Proof: Assume that Φ is unsatisfiable. That means there is no belief state S such that *ASK*$(S,\varphi_i) = yes$ for all $1 \leq i \leq n$ and such that *ASK*$(S,\psi_i) = no$ for all $1 \leq i \leq m$. Since the negative faithfulness constraint holds, there must be no state S such that *ASK*$(S,\varphi_i) = yes$ for all $1 \leq i \leq n$ and *ASK*$(S,\neg B(me,\psi_i)) = yes$ for all $1 \leq i \leq m$. Therefore, the sequence $\varphi_1,\ldots,\varphi_n,\neg B(me,\psi_1),\ldots,\neg B(me,\psi_m)$ is not acceptable in S_0.

A proof $\Phi \vdash \perp$ can be constructed as follows: from each literal $\neg B(\alpha,\psi_i)$, the rule of negative introspection licenses the conclusion $B(\alpha,\neg B(me,\psi_i))$. Then, from

$$B(\alpha,\varphi_1),\ldots,B(\alpha,\varphi_n),B(\alpha,\neg B(me,\psi_1)),\ldots,B(\alpha,\neg B(me,\psi_m)),$$

the rule of negative simulative inference licenses the conclusion $\perp$, since the sequence $\varphi_1,\ldots,\varphi_n,\neg B(me,\psi_1),\ldots,\neg B(me,\psi_m)$ is not acceptable in S_0. □

The completeness result shown in Section 3.4.2 (for the set of rules including the original simulative inference rule) said that if a theory with no universal quantifying-in was unsatisfiable, then it had a refutation proof. We have not attempted to prove a similar property for a set of rules in which the original simulative rule is replaced by a rule of negative introspection. However, Theorem 14 has the corollary that if the new rules are augmented with a set of rules that is complete for ordinary first order

logic, the resulting set is refutation complete for the variant of our logic that prohibits quantifying-in entirely.

The above completeness result suggests that the combination of negative simulative inference and negative introspection replaces, in a sense, the original positive simulative inference rule. We now show that to be true, by showing that the following rule, which is similar in form to the original positive simulative inference rule, can be derived from negative simulative inference, negative introspection, and *reductio ad absurdum*:

$$\frac{B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)}{B(\alpha, \psi)}$$

if *ASK*(*TELL*$(S_0, \varphi_1, \ldots, \varphi_n, \neg B(me, \psi)), \chi) = no$ for some $\chi \in \{\varphi_1, \ldots, \varphi_n, \neg B(me, \psi)\}$

The premises and conclusion of this rule are the same as those of the original positive simulative inference rule. Only the *ASK*/*TELL* condition under which it applies is different. Theorem 15 shows the derivation of the new inference rule.

Theorem 15 *For any term α and sentences $\varphi_1, \ldots, \varphi_n$ and ψ, if*

$$\textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n, \neg B(me, \psi)), \chi) = no$$

for some $\chi \in \{\varphi_1, \ldots, \varphi_n, \neg B(me, \psi)\}$, then there is a proof

$$B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n) \vdash B(\alpha, \psi)$$

using the rules of negative simulative inference, negative introspection, and reductio ad absurdum.

Proof: The proof can be constructed as follows: begin with the premises

$$B(\alpha, \varphi_1), \ldots, B(\alpha\varphi_n),$$

and make the assumption $\neg B(\alpha, \psi)$ for the purposes of deriving a contradiction. From the assumption, the negative introspection rule licenses the conclusion $B(\alpha, \neg B(me, \psi))$. From this conclusion and the original premises, the rule of negative simulative inference licenses the conclusion $\perp$, since the sequence $\varphi_1, \ldots, \neg B(me, \psi)$ is not acceptable. Discharging the assumption, we can conclude $B(\alpha, \psi)$ by *reductio ad absurdum*. Here is the proof in tabular form:

1.	$B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)$	premises
2.	Assume $\neg B(\alpha, \psi)$	
3.	$\quad B(\alpha, \neg B(me, \psi))$	2, negative introspection
4.	$\quad \perp$	1,3, negative simulative inference
5.	$B(\alpha, \psi)$	*reductio ad absurdum*

□

3.7 Summary of Mathematical Results

The most important technical results are those related to the soundness of positive and negative simulative inference. The positive simulative inference rule is sound for any belief machine that satisfies constraints C1–C4. The negative simulative inference rule is sound under constraints C1, C2, and C4 (C3, monotonicity, is not required). Not only are the constraints jointly sufficient for the soundness of the simulative inference rules, but some of them are also necessary.

With the indexical constant me in the language, negative introspective belief machines are possible. Non-trivial negative introspection is a violation of the monotonicity constraint, which means that positive simulative inference is not sound for such machines; but the negative simulative inference rule doesn't require monotonicity for its soundness, and can be combined with a rule of negative introspection to yield a new form of positive simulative inference which is sound. Essentially, negative introspection allows a simulation to be given explicit information about the absence, as well as the presence, of beliefs, so simulative inference can be monotonic even when using a nonmonotonic belief machine.

We listed some other inference rules, gave various soundness proofs for them (some are only applicable in restricted environments, or in the absence of the indexical me), and proved that the entire set of rules is complete, for the language without the indexical and without universal quantification into positive belief contexts.

We also considered a number of classical axioms of belief, and discussed the classes of belief machines for which each is valid. Some are valid for reasonable and interesting belief machines, while others are idealizations that aren't satisfied by any machines, or only by trivial ones.

4 Implementation

The development of the mathematical model we have presented was of course motivated by the desire to add simulative inference about belief to a reasoning system in a principled way. In this chapter, we explore some of the practical considerations involved in doing so, by describing our project to add simulative inference to EPILOG, a knowledge representation and reasoning system designed to support natural language processing.

We begin by introducing the EPILOG system, as it existed before this project began, and we use constraints C1–C4 to determine how simulative inference can appropriately be added to the system. Then we discuss the problem of implementing simulative inference *efficiently*, an important issue which is not addressed by the theoretical work. In the course of the implementation, we discovered that EPILOG has some properties that make simulative inference difficult to implement in an efficient way; these are general architectural properties which are not unique to EPILOG, but are shared by other AI systems. The problem is also not specific to simulative inference. It will arise for any reasoning technique that involves keeping track of the inferential path by which a formula was derived, which (as we will show) includes simulative inference, as well as other techniques such as truth maintenance for belief revision, probabilistic reasoning, and explanation generation. We describe how we worked around the problem in EPILOG, and discuss how general this solution might be.

4.1 EPILOG

We will first describe the features of EPILOG which are relevant to this work. A more thorough description can be found in [Schubert and Hwang, 2000; Hwang and Schubert, 1993] and at `http://www.cs.rochester.edu/research/epilog/` . The system is also available at that address.

Episodic Logic (EL) is a knowledge representation language intended for use as the semantic representation in a natural language processing system. It is a very expressive logic, designed to make the mapping between natural language and the logical form

straightforward. It combines features of Montague grammar, situation theory, DRT, and other formalisms, with the aim of being as expressive as natural language but with a precise, unambiguous, model-theoretic semantics.

EPILOG is a computer program that performs inference in EL. Rather than being a general-purpose theorem prover, the system is tuned to perform efficiently certain kinds of inference, intended to approximate the inferences a human would make. The system has been demonstrated at story understanding tasks, in which a fragment of a narrative or newspaper article, translated into EL, is entered, and then the system answers questions about the contents. Typically, first a body of *world knowledge*, common-sense facts about the world, and *meaning postulates*, axioms that describe relationships between the meanings of various words, is entered. (We will sometimes use the term "world knowledge" broadly to encompass meaning postulates as well; the distinction is not relevant for our current purposes. Both are background information that all competent adult speakers of English can be assumed to know.) Next, the story itself is entered. As each sentence is entered, the system performs some *input-driven inference*, explicitly drawing conclusions which were implicit in the text, given the world knowledge and meaning postulates. These are intended to be the conclusions which would occur spontaneously to a human upon reading the story. After the story has been entered, the system can answer queries. The inference involved in answering queries is called *goal-driven inference*. Again, the aim is not for the system to be able to prove arbitrary theorems of EL, but for it to be able to answer the sorts of questions that a human could answer after reading the same story.

To this end, the system has a variety of different inference techniques and knowledge representations at its disposal. The primary knowledge representation is EL, the same language in which the input and queries are expressed. Sentences of EL are indexed by the predicates and individual constants that occur in them, for efficient retrieval of relevant sentences for inference, and the system has natural deduction style inference rules that operate on sentences of EL. In addition to this general-purpose reasoning facility, the system contains a number of *specialists*, modules that use special-purpose techniques and knowledge representations to accelerate particular kinds of reasoning. For example, there are specialists for handling temporal reasoning, taxonomic (type subsumption and exclusion) reasoning, reasoning about sets, and reasoning about equality. All information is initially entered as sentences of EL, and is stored in that form; but when a sentence is entered, if it involves a kind of information in which one of the specialists has indicated interest, it is also entered into the specialist, which may store the information in a different, non-sentential form. For example, the temporal specialist builds a graph structure, in which nodes represent time points and links represent before/after relationships.

We will describe a few more details of the system as they become relevant to the following discussion.

4.2 EPILOG as a Belief Machine

The practical importance of the theoretical work we have presented is that it identifies a class of inference algorithms to which simulative inference can be added in a principled, demonstrably correct way. We will now use EPILOG as a case study in applying the model in the real world.

The first question is whether the system can be modeled as a belief machine. EPILOG does essentially work according to the *TELL*/*ASK* model, and is guaranteed to halt on any input, but there is one substantive extension: in addition to yes/no queries, it can answer "wh-queries," such as "Who killed JFK?". This ability does not cause a problem for the application of simulative inference—if we simply ignore it, never making wh-queries to a simulation, then the simulation can be viewed as a belief machine.

Doubts about the applicability of the model are also raised by the fact that the syntax and semantics of Episodic Logic are quite different from the mostly ordinary first-order logic we used in Chapters 1 and 3. Our most important result, the soundness proof for simulative inference, doesn't depend on the details of the language of belief—it simply treats sentences as units, without regard to their internal structure—and so is also applicable in this richer language. For other results, particularly ones that involve quantifying-in, it is not so clear. More work would be required to reconcile our logic of belief completely with EL.

For present purposes, let us concentrate on the soundness of simulative inference. Having accepted that the part of the system being used for simulation can be modeled as a belief machine, we can next ask whether it satisfies, or can be made to satisfy, constraints C1–C4.

Monotonicity: One feature of Episodic Logic not found in ordinary first-order logic is the probabilistic conditional—in EL, one can express rules such as "If x is a person, then with probability $\geq .95$, x lives in a building." In EPILOG's internal representation, each sentence is labeled with a lower bound on its subjective probability. Input facts are given a probability of 1, but sentences derived via probabilistic rules can have lower probabilities. It is possible for the system to assign non-zero probabilities to both a sentence and its negation. When a query φ is posed, the system answers "yes" if the probability of φ is greater than the probability of $\neg\varphi$, and "no" otherwise. Probabilities can change in light of new information. If the probability of $\neg\varphi$ is initially lower than that of φ, but is later increased past that of φ, then effectively φ is believed and then retracted; this retraction is not necessarily accompanied by the retraction of any of the premises on which belief in φ was founded. This is clearly a violation of the monotonicity constraint. Therefore, for simulative inference to be sound, we must avoid entering sentences with probabilistic conditionals into simulations.

The monotonicity constraint is perhaps the most difficult to accept, because non-monotonicity is such a prominent feature of human belief. At the same time, of all the

constraints, monotonicity is the most obviously necessary for the soundness of simulative inference. The natural conclusion is that in practice, simulative inference must rarely be sound—we must make defeasible inferences about others' beliefs, based implicitly on defeasible assumptions about what they believe and don't believe. While it is somewhat disappointing (though not surprising) to have to make this concession, it does not negate the significance of our results. To use a defeasible rule in a principled way, one must understand for what reasons it might fail. As we are about to see, constraints C1–C4 can illuminate several different potential sources of error for simulative inference in a system. One must consider each of them, and decide whether it is a legitimate source of defeasibility, as with this particular violation, or a true mistake, the result of applying an inference rule inappropriately.

The propagation of changes in probability is EPILOG's only way of dealing with inconsistent input, so without probabilistic reasoning the system is radically monotonic: not only is every acceptable sequence monotonically acceptable, as required by the constraint, but in fact *every* sequence is monotonically acceptable. Once φ is believed, it will always be believed, even if $\neg\varphi$ comes to be believed as well.

Acceptable basis: This constraint says that for every possible belief set, there must be an acceptable sequence of inputs that induces that belief set. With probabilistic reasoning disallowed, the acceptable basis constraint is trivially satisfied, because every input sequence is acceptable. (It might be the case that EPILOG has unreachable states, *i.e.* states which can't be induced by *any* input sequence; but if so, there is an equivalent abstract belief machine that has no such states.)

Closure: The closure constraint requires that *TELL*ing the belief machine something it already believes does not change the belief set. This constraint does not hold if we use EPILOG's full-strength question answering function as the simulation's *ASK*. The termination of goal-driven inference is guaranteed by imposing a limit on the number of inference steps. It violates the closure constraint in essentially the same way as the example machine on page 6, which imposed a time limit on question answering. However, EPILOG's query function has an optional argument which specifies the effort level to be used. With an effort level of zero, the system only assents to sentences which have previously been explicitly stored, as a result of being entered or inferred, and to sentences the specialists can confirm using constant-time inference methods. At effort level one, it will break a goal into subgoals and attempt to verify each of them independently, by the methods used at level zero or by further decomposition, but will not apply inference rules that combine multiple facts or rules to generate new subgoals. At levels two and higher, inference chaining is performed, and limited by a threshold, and therefore the closure constraint is violated. So for simulative inference to be sound, we must define the *ASK* function to be EPILOG's query function with the effort level set to one.

As we pointed out in Section 2.5.4, the belief machine need not be seen as describing an agent's entire reasoning capability. In this case, we needn't prohibit EPILOG from using effort levels two and higher in answering a user's queries; it is only simulations of other agents' beliefs that must be restricted to the low-cost query method.

This restriction does not mean that no inference chaining will take place in simulations. In addition to the inference performed by the specialists, which potentially uses many input facts simultaneously, there is the input-driven inference mechanism. Input-driven inference is not restricted by a depth threshold (we will describe shortly the mechanism by which it is constrained), so forward inference needn't be disabled to satisfy the closure constraint. This state of affairs is quite intuitively plausible, and in keeping with the conception of belief we have described. An agent believes that which it can verify without reflection, but learning new information may trigger reflection, and what is inferred as a result of that reflection is then believed.

If the monotonicity constraint had not already forced us to disallow probabilistic reasoning in simulations, the closure constraint would do so. If probabilistic inference is allowed, then the system can assign a sentence φ a probability less than 1. As long as $\neg\varphi$ is not assigned a higher probability, the system will answer "yes" to the query φ. But if φ is then entered as a new input sentence, its subjective probability will be raised to 1. The probability increase might propagate through the system, with the possible result that some previously disbelieved sentence comes to be believed. This would violate the closure constraint.

Commutativity: In certain well-defined situations, EPILOG is sensitive to the order in which information is entered. One is that if the system's world knowledge is augmented after the processing of a story has begun, then some inferences may be missed that would have been made if all of the world knowledge was available at the beginning of the story. The other is that if an assertion about an entity's type is made after other facts about that entity have been entered, then some inferences may be missed that would have been made if the type had been known when the entity was first mentioned. These order dependencies are artifacts of the way the inference mechanisms are implemented, not intended properties—although certain inferences are only made when the premises are presented in a particular order, we would like to attribute those inferences to others regardless of what order we believe them to have learned the premises. Therefore, just as the EPILOG manual warns the user about the order dependencies so that he can take care to enter information in the most effective order, we can simply implement simulative inference so that it enters information into simulations in the most effective order.

In the actual implementation, we have not taken any action to ensure this, leaving to the user the burden of presenting sequences in the most useful order (as was already the case). If the user is able to enter all of the world knowledge before the narrative begins, then it will be available at the beginning of each simulation, as desired (this will

be explained in Section 4.3); and just as the user previously had to inform the system of an entity's type before providing any other information about it, he now must tell the system what type an agent believes an entity to be before reporting what else the agent believes about that entity.

A second threat to commutativity comes from a criterion that can be used used to limit the extent of input-driven inference. The system maintains an "interestingness" value for each predicate, term, and formula, and input-driven inference can be truncated when the interestingness of the conclusions being drawn goes below a threshold. Changes in the way interestingness is calculated are planned, but currently it works as follows. The interestingness of a predicate is fixed, part of the system's lexical knowledge. The interestingness of a term starts at a particular value, and then increases as formulas involving the term are learned. The interestingness of a formula is calculated when the formula is first entertained, and remains fixed thereafter; the calculation involves the interestingness of the predicates and terms occurring in the formula, and, if the formula was inferred from others, is augmented by a fraction of their interestingness. The component of a formula's interestingness inherited from its inferential ancestors is the potentially problematic part. Suppose ψ has a low intrinsic interestingness, but is derivable from φ, which has a high interestingness. Consider first the case in which φ and then ψ are entered. When φ is entered, input-driven inference yields ψ, which inherits some of φ's interestingness. If this inheritance raises ψ's interestingness above the threshold, then input-driven inference will continue, perhaps deriving χ. Subsequently entering ψ will have no effect, since it was already believed. Now consider the case in which ψ and then φ are entered. When ψ is entered, its intrinsic interestingness is found to be below the threshold, so no input-driven inference is attempted. Next, φ is entered. ψ is derived, but is found already to be believed (and recall that a formula's interestingness remains fixed after the formula is first entertained), so no further inference is attempted. With the inputs presented in this order, the conclusion χ is not reached.

In the current version of EPILOG, the interestingness threshold is set to zero by default, meaning that input-driven inference continues until all possibilities are exhausted. This is presumably because the metric currently being used for interestingness was found to be ineffective (we have not communicated with the original author about this). Inference is guaranteed to halt eventually, thanks to checks other than the interestingness threshold, but the consequences of every new conclusion are explored, regardless of how uninteresting it is. With the threshold set to zero, we needn't worry about the commutativity constraint being violated, but the system's ability to handle large knowledge bases is undoubtedly impaired. There are no ill effects when using knowledge bases as small as those in our test example (see Section 4.6.2), but with larger knowledge bases the input-driven inference would take unacceptably long, drawing untold numbers of uninteresting conclusions. As EPILOG's interestingness metric is refined in the future, we can take constraints C1–C4 into account to ensure that it is appropriate for use with simulative inference.

In this case study, we have used the formal results of Chapter 3 to identify a number of ways in which simulative inference in EPILOG might yield incorrect results, and the solutions we have chosen illustrate the variety of ways in which such problems might be managed. One problem, the violation of the closure constraint, can actually be corrected by implementing simulative inference properly. For the problems of commutativity introduced by the interestingness criterion, what we have discovered will inform future refinement of the criterion. In the case of the system's known sensitivity to the order of its inputs, we simply presented an argument that the problem will not manifest itself in normal use. Finally, in the case of the system's nonmonotonicity, we showed how the problem can be avoided (by prohibiting the entry of probabilistic information into simulations), but had to acknowledge that this solution is not satisfying. For this reason, simulative inference ultimately should be considered a defeasible method, rather than a sound one; but by pointing out and dealing with the various other reasons for unsoundness, we ensure that this will be done in a principled way, not allowing incorrectness to hide behind legitimate defeasibility.

4.3 Efficient Implementation

There is an enormous amount of "common sense" information that every competent human knows, and that must be imparted to a computer system if it is to emulate human reasoning. People know that things fall downward when dropped, that elephants are bigger than ants, and that bread is a kind of food;[1] and having this knowledge is crucial for a system to perform human-like reasoning or to communicate with humans. All of this information is not only known by every competent person, but is "common knowledge" among us. This means that everyone knows that everyone knows it, and knows that everyone knows that, *etc.*. It is because we have so much common knowledge that we can rely so much on inference for the success of our communication.

In order for EPILOG to emulate human reasoning, then, it must have a large amount of world knowledge, and it must also attribute possession of this knowledge to those about whose beliefs it reasons. This means that the information available to a belief simulation will be not just a few sentences that the system has explicitly identified as beliefs of that agent, but also a vast number of sentences which the system knows that everyone believes.

It would be a tremendous waste of storage space to duplicate all of this information in every simulation. To build a reasonable implementation of simulative inference, we must store the common world knowledge only once in the system, and allow all simulations, as well as the system's own reasoning, access to it. At the same time, there must be a way for the system to hold some beliefs without attributing them to others, and vice versa.

[1] All of these "facts" are defeasible in the right circumstances; people know this, too.

We have done this by introducing structures called *environments* into EPILOG. An environment is an encapsulization of all of the various data structures the system uses for storing knowledge. A single environment contains an instance of each such data structure that was present in the original EPILOG, but now many environments can be maintained simultaneously in one running instance of EPILOG. One environment stores the common world knowledge that the system uses for itself and also attributes to others; another stores the system's own beliefs that it doesn't attribute to others; and for each agent that the system simulates, there is another environment for beliefs the system attributes specifically to that agent.

We have modified EPILOG's inference mechanisms so that information from multiple environments can be used simultaneously. At any given time, only some environments are accessible. One is designated as the primary environment, and there may also be any number of secondary ones (we currently use only one secondary environment at a time, always the common world knowledge environment, but we envisage more complicated schemes in which an environment stores beliefs attributed not to everyone, but to every member of some group; in such a scheme, several secondary environments might be accessible simultaneously). Knowledge from any of the accessible environments may be used in inference, but if changes are to be made as a result of new information, only the primary environment can be modified. Ordinarily, the system uses its own private environment as the primary one, and the common world knowledge environment secondarily. When simulating another agent's beliefs, the system temporarily makes its own private environment inaccessible, and makes the simulation environment for that agent primary. The common world knowledge environment can also be made primary, in order to add new knowledge to it.

The designers of other systems have also used the partitioning of a knowledge base into environments (also called *contexts* by some authors) to solve various other, independent problems. In Cyc [Lenat and Guha, 1990], propositions are grouped into environments based on common implicit assumptions, as a way of maintaining a kind of consistency across a large knowledge base; in many systems, going back at least as far as CONNIVER [McDermott and Sussman, 1972], a hierarchical arrangement of environments allows the system to reason simultaneously about different possible scenarios; and in Rhet [Allen and Miller, 1991], an environment mechanism is used to implement simulative inference as well as the sort of hypothetical reasoning introduced by CONNIVER.

As we described in Section 4.1, EPILOG consists of a general-purpose reasoning facility, which stores information as sentences of EL, and a number of specialist modules, some of which use their own special-purpose data structures to speed up particular kinds of reasoning. The data structure that supports the general-purpose reasoning is simply a hash table containing sentences of EL, indexed by the most important predicates and terms they contain. Partitioning this data structure into multiple environments is straightforward: each environment contains one of these hash tables. When retrieving sentences for possible use in inference, when the system previously would have

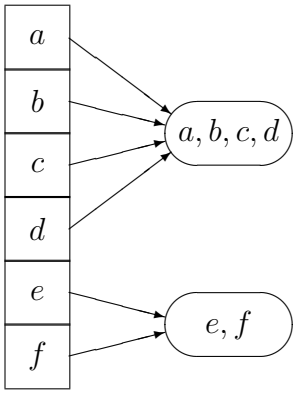
looked in a single hash table and collected a list of candidate sentences, it now looks in the hash tables of all accessible environments. As long as the number of accessible environments is small (we currently use no more than two, one for agent-specific knowledge and one for common world knowledge), the extra cost is negligible.

For some of the specialists' data structures, it is not so simple. For example, consider the representation used by the equality specialist, which uses the well-known "union-find" algorithm. In this representation, each equivalence set of terms is represented by one set object. Each term has a pointer to the structure denoting the set of which it is a member, and each set structure contains a list of all of its members. The question of whether two given terms are equal can be answered in constant time (assuming a perfect hash function), by finding which set each term belongs to, and comparing to see if they are the same. The set of all terms equal to a given term can also be found in constant time. In this representation, the set of sentences

$a = b$
$b = c$
$c = d$
$e = f$

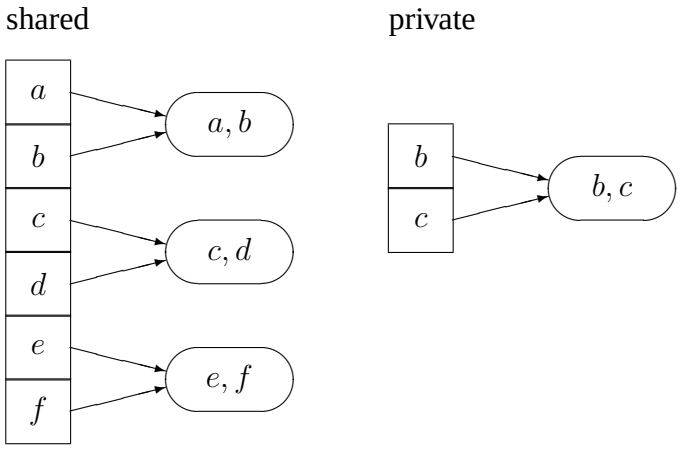
gives rise to the structure shown in Figure 4.1.

Figure 4.1: A union-find data structure



The problem is that for this data structure, unlike the simple hash table of sentences used by the general-purpose reasoning mechanism, it is not obvious how to partition it into one structure for beliefs shared by all agents and another for each individual's private beliefs. Say the identities $a = b$, $c = d$, and $e = f$ are common world knowledge, but the identity $b = c$ is a belief that the system holds but does not attribute to others. One possibility is to build a separate graph for each environment. The same propositions would then be represented as in Figure 4.2.

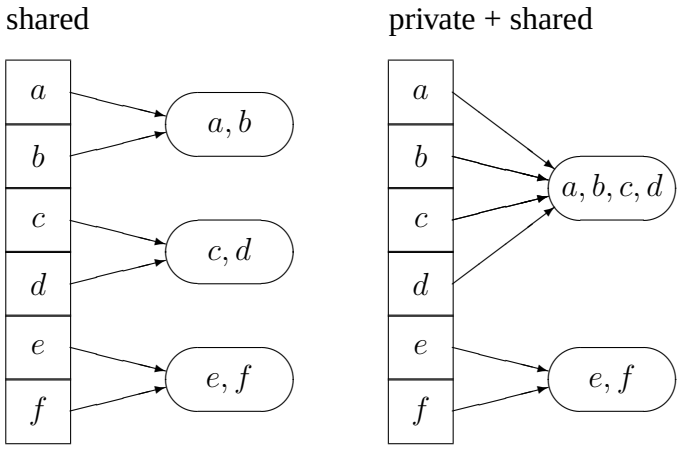
Figure 4.2: The structure split between two environments



But then when reasoning normally, *i.e.* with both environments simultaneously, the system can no longer read the equality $a = d$, for example, directly from the special-purpose representation. The graph query algorithm could of course be modified so that it could discover such inter-environment transitivities. It would then be less efficient than the original algorithm for queries that involve information from both environments, but would retain its efficiency for purely intra-environment queries, so the special-purpose representation might still serve some purpose.

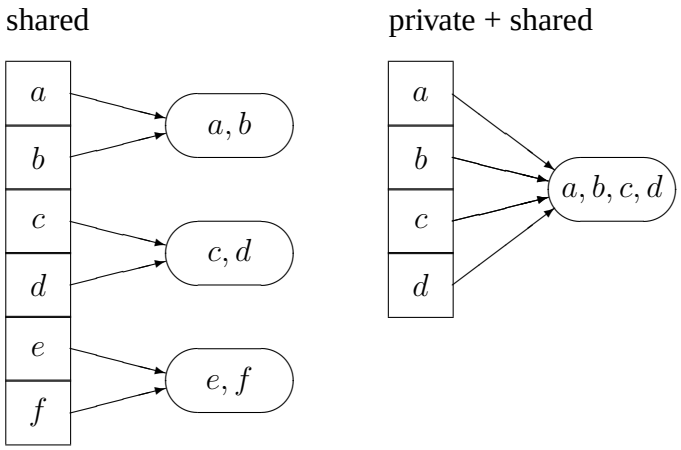
Since the private knowledge base is always used in conjunction with the shared one, never by itself, the ability to answer queries that depend on information from both environments could be regained by maintaining another environment, representing the combination of the shared and private environments. We would then have the structures shown in Figure 4.3.

Figure 4.3: Duplicating shared knowledge in the private environment



However, the shared knowledge base in a practical system is likely to be very large, so it would be undesirable to duplicate all of that information in the system's private environment, and in all of the environments the system uses for simulating other believers. Therefore, as a further refinement, instead of explicitly storing the entire contents of the combined private+shared environment, one could build a structure that represents only the differences between the combined environment and the shared one. The resulting structures would be as shown in Figure 4.4.

Figure 4.4: Duplicating only modified information



Note that the combined private+shared structure contains no information about e or f, since that information would have been identical to what can already be found in the

shared environment. When reasoning with both the private and shared environments, one retrieves an equivalence class by looking first in the combined environment, and then in the shared environment if it was not found.

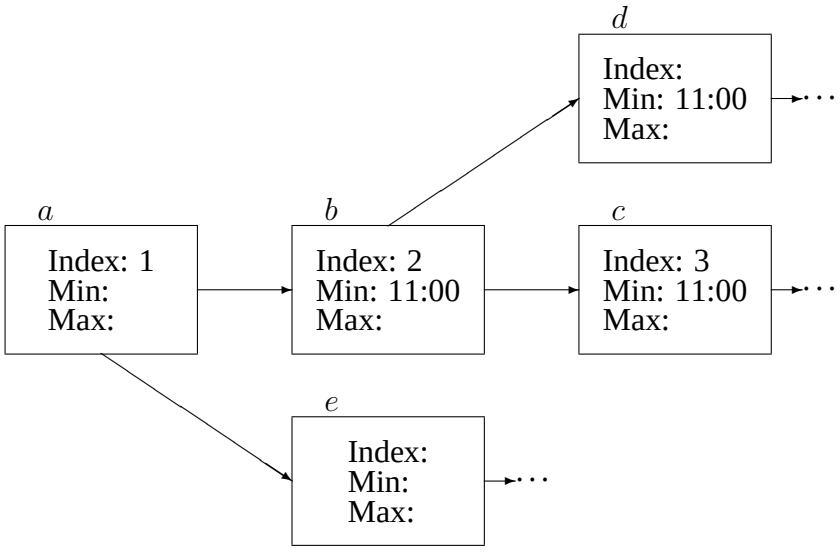
This last strategy, of representing a particular agent's beliefs by storing the differences between those beliefs and the larger set of common beliefs, is the main technique we have used to adapt special-purpose representations to multiple-environment reasoning in EPILOG. If the system (or, likewise, a simulation) believes nothing but what is common knowledge, then its primary environment is empty. This will typically be the initial state of the system when it begins processing a new narrative, and of a simulation when nothing has yet been entered about an agent's private beliefs. In this state, the system answers queries using only the structures in the shared environment. When new information is entered, it is first stored in the general-purpose sentential representation, in the current primary environment (which is the system's private environment ordinarily, or a simulation environment during a simulation). If no specialists are interested in the new information, then the process is finished. If a specialist decides to store the information in its special-purpose representation, then it finds all of the parts of its data structure that need to be modified to reflect the new information. If any of these are in the secondary environment, then copies are made in the primary environment before the changes are made. When a piece of the data structure is duplicated, the version in the primary environment takes precedence.

This strategy yields efficiency similar to that of the single-environment case under the assumptions that the changes to the data structure required to reflect the assertion of new information are relatively local, *i.e.* that assertion of a new literal requires modification of relatively few nodes of the graph. This assumption is valid for the equality representation (the number of members per equivalence class is typically small compared to the number of equivalence classes), and for some of the other representations used in EPILOG, but not for all of them. In particular, the locality assumption does not hold for the representation used by the temporal reasoning specialist, which accelerates reasoning about the relationships between events in time. In this representation (but ignoring some of the details for the moment), the information

before(a, b)
before(b, c)
before(b, d)
before(a, e)
before(11:00,b),

plus more information about events that take place after c, d, and e, might be represented by the structure shown in Figure 4.5.

Figure 4.5: A temporal graph



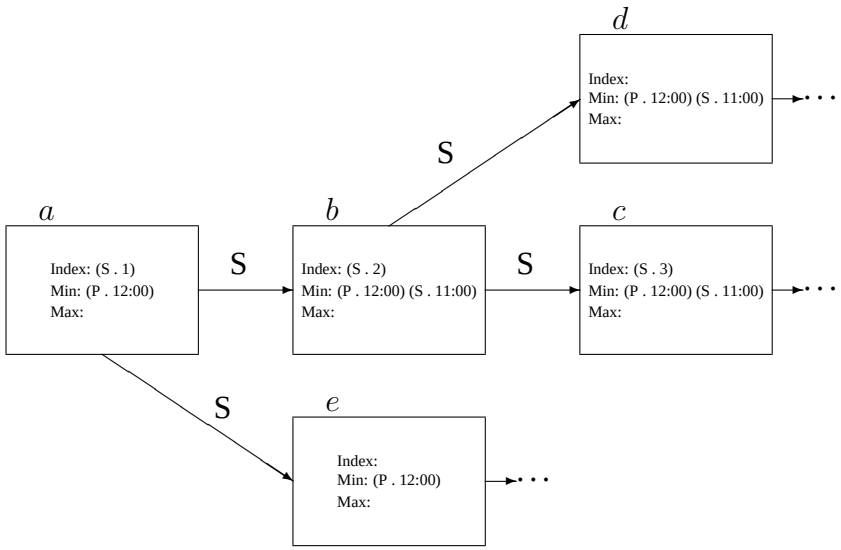
Note that the available information about the times of these five events only induces a partial ordering, but we do have a total ordering for the subset $\{a, b, c\}$, and those three nodes of the graph have been given integer indices to indicate their relative positions in that ordering. A query such as before(a, c) can be answered quickly by simply comparing the labels of the two nodes: since the index of a is lower than the index of c, before(a, c) is true. Answering this type of query would take only a constant amount of time no matter how long the chain of events between a and c happened to be. Also note that information about absolute time (assume for this simple example that the date is implicit) has been propagated through the graph where possible: since b is after 11:00, and c and d are after b, the graph shows that c, d, and e are all after 11:00.

To see how non-local the effects of a single change can be, consider the effects of adding the assertion before(12:00,a) (meaning that 12:00 is before a, or equivalently that a is after 12:00) to the above graph. Every node in the tree rooted at a will be modified to reflect the fact that they all occurred after 12:00. Now, if the information that gave rise to the original graph is all common knowledge stored in the shared environment, but the new belief before(12:00,a) is not to be attributed to others, then the entire tree will need to be copied from the shared environment into the private one, so that every node can be modified there.

This kind of large-scale duplication is what we are attempting to avoid. The solution we have used is simply to block the propagation of absolute time information between environments. The cost of this decision is that some "before/after" queries that could formerly be answered in constant time, by comparing absolute times, will now be answerable only by performing a graph search.

We have considered, but have not yet tried implementing, another alternative, namely to dispense with the strategy of duplicating segments of the shared environment in the private environment before modifying them. Instead, there would be a single graph combining the information from all environments, and each field of each node would contain, rather than a single value, a list of values, each labeled with the name of the environment in which it holds. The example graph would become the one shown in Figure 4.6, where values (including links) to be considered part of the shared environment are labeled 'S', and those in the private environment are labeled 'P'.

Figure 4.6: A single graph containing both private and shared knowledge



The disadvantage of this technique is that retrieving the value of a field of a graph node now requires a search through a list, and is therefore no longer a constant-time operation. As the number of different environments becomes large, this could become significant; but if the number of agents being reasoned about remains relatively small, or if the beliefs we attribute to different agents tend to involve mostly disjoint sets of nodes, then this approach may be successful.

A hybrid of the two techniques is also conceivable: one could use the copy-on-write technique when modifying some fields (those for which modifications tend to have localized effects), and annotated field values for others (ones for which modifications tend to propagate widely).

4.4 The Generality of the Efficiency Problem

In addition to the idea of partitioning knowledge into multiple reasoning environments, we initially considered an alternative scheme. In this scheme, the system would maintain its private beliefs and common world knowledge in the same data structures, just as it was done before the implementation of simulative inference. To perform a simulation, it would first temporarily assume as its own, in addition to all of the existing private and shared knowledge, any beliefs that were attributed specifically to the agent being simulated. Then it would attempt to answer the simulative query using its ordinary query-answering mechanism. If the query was successful, the system would examine each of the premises used in the proof (this information is returned by the query mechanism when a proof is found), and determine whether belief in that premise could be attributed to the agent. If the agent was thought to believe all of the premises, then the simulation would be considered to have answered yes, and the simulative inference would go through. When the simulation was finished, the agent's beliefs that the system had temporarily assumed as its own would be retracted.

Conceptually, this scheme has several advantages. It would be simpler to implement, since the complexity of partitioning the knowledge into multiple environments, and reasoning with information from multiple environments simultaneously, could be avoided. It would require each input sentence to be annotated with the source from which the system learned the sentence, so that it could be determined whether the agent being simulated had access to that source; but this is easily implemented, and it might even allow more fine-grained distinctions between information sources. Instead of just a single mass of world knowledge labeled as "common knowledge," one could just as easily have the system's knowledge come from arbitrarily many different information sources, and have the system reason explicitly about whether a particular agent had access to each source. This wouldn't be practical using the environment method, since with that many environments accessible simultaneously, the inference mechanism would be much less efficient.

The problem with this scheme is that while EPILOG's question answering mechanism ordinarily does return a list of premises that were used in a proof, if any specialists were involved in the proof, the premises on which they based their answers are not included in the list. Once a set of sentences has been entered into a specialist's data structure, there may be no way to identify the contribution of each individual sentence to that structure, and no way to list the sentences that are responsible for the specialist's answer to a particular query. This is precisely the problem with which we dealt in the previous section, and the alternative approach to simulative inference we are currently considering does not suggest any alternative solutions to the problem.

What is particularly interesting about the alternative approach is that it brings out a similarity between simulative inference and a set of other, more widely studied inference techniques. It shows that what we are doing to support simulative inference

can be viewed as a sort of reason maintenance. We need to keep track of the system's reasons for holding each of its beliefs, in order to know whether another agent might believe the same thing. The obstacle we dealt with in the previous section is a conflict between two different knowledge representation techniques: reason maintenance, and special-purpose knowledge representations that accelerate specific kinds of inference.

In addition to simulative inference, there are several purposes for which a system might keep track of its reasons for believing things. One is for generating explanations: if a system keeps track of the inferential paths it follows, then it can present those paths to a user as an explanation of its conclusions. Another is to support belief revision: if a system keeps track of all of the conclusions it has drawn from a particular premise, then if it later needs to retract that premise, it can also be sure to retract any conclusions that are no longer justified. Probabilistic reasoning can involve a variation on this theme: a system that labels its beliefs with epistemic probabilities might keep track of its evidence for each formula, so that if its degree of confidence in that evidence changes, the change can be propagated to the conclusion.

We are not the first to encounter this problem. For example, Rhet [Allen and Miller, 1991] uses the union-find algorithm as well as multiple reasoning environments, and combines them in a way that is essentially similar to what we have done. The contribution of this section, therefore, is to point out the broad generality of the problem. The two knowledge representation techniques we have discussed have each been written about extensively in isolation, motivated by seemingly compatible considerations, and presumably with the understanding that an eventual comprehensive inference system would make use of both; we have not previously seen it documented that the combination of the two techniques is problematic.

Our particular way of dealing with the problem is based on some very specific assumptions, among them that the body of common world knowledge is much larger than any other category, that the number of categories of information being used at one time will be small, and that there will be no need to retrieve a list of sentences that are believed by an agent but are not common knowledge. These are reasonable assumptions for our application, but are likely not to be for others.

4.5 Current State of Implementation

The development of EPILOG is an ongoing project. A form of simulative inference is currently implemented and working, but there are many extensions and improvements that could be made.

Before this work began, the system's state was stored in a number of global variables. Of these, some could be viewed as storing a part of the system's knowledge base (KB), and others part of the inferential state. After our modifications, the locations that store KB state are no longer global variables; instead, they are fields of an environment

structure. Any number of environments may exist in the system simultaneously. There is a global variable that picks out the current primary environment which is initially set to the system's private environment.

In addition to fields that store KB state, an environment structure contains a list of other environments accessible from that one. All of the environments accessible from the primary environment, and the environments accessible from them, *etc.*, are treated as secondary environments (currently we only use a single secondary environment, the common world knowledge environment, which has no accessible environments of its own, but this limitation is not built into the system). This means that the system cannot be asked to reason with an arbitrary combination of environments; it can only be instructed to use a single environment as its primary environment, and the set of secondary environments is determined by the primary one.

We have made the simplifying assumption that primary environments only augment the knowledge contained in their secondary environments, rather than contradicting or blocking the inheritance of any facts. This is quite a substantial assumption—it means that an agent who believes all but one of the (potentially millions of) facts in the shared environment cannot be modeled, unless by making the shared environment inaccessible to that agent's simulation, and duplicating the entire shared environment, less the disbelieved sentence, in his private environment. To relax this assumption would require a facility for retracting an arbitrary sentence from the knowledge base, something EPILOG does not currently have. We also assume that the contents of an agent's secondary environments do not change after knowledge has been entered into the primary one. Relaxing this would be useful in certain scenarios, but would require more expensive computation each time information was entered into a primary environment.

The bulk of the work of the implementation involved adapting the inference mechanism to use environments. Wherever the mechanism formerly referred to a global variable, it must now use a field of the environment structure, and it must consider not a single environment, but all accessible environments. Because of the nature of EPILOG, making this conversion involves a number of separate and independent tasks: the general-purpose inference mechanism and each of the specialists must be converted in turn.

The change for the general-purpose inference mechanism has been completed, and was quite straightforward. Where the mechanism once looked in a single hash table for relevant sentences to use in inference, it now looks in the hash table of each accessible environment, and collects sentences from all of them.

For each of the specialists, a different sort of change is required. Some of the specialists don't maintain any state of their own, so they require no modification. Others must be modified to store their state in a field of an environment, rather than a global variable. This simple modification has been made to all of the specialists. The next step is to modify a specialist's inference mechanism to use multiple environments simultaneously; this has been completed for only some of the specialists, namely those for

reasoning about set-theoretic relations, equality, and time. The remaining specialists which maintain some kind of state and have not been modified to use multiple environments are those for type subsumption, part-of relations, color, and numerical relations. For the time being, these specialists have been instructed always to use the common world knowledge environment as if it were the primary environment. Anything entered into these specialists is treated as common knowledge, available to all simulations and to the system itself.

With this infrastructure in place, simulative inference has been implemented by adding a specialist module that handles the predicate `believe`. Whenever a sentence of the form `believe(`α`,`φ`)` is asserted, the belief specialist is notified. If this is the first belief that has been attributed to α, the specialist creates a new environment, adds the common world knowledge environment to its list of accessible secondary environments, and stores it in a hash table indexed under the name α; otherwise, it retrieves the existing environment for α from the hash table. Then it enters the sentence φ into the environment by temporarily making the environment the primary one, and then calling the same entry function that a user would call to enter information into the system. Input-driven inference occurs just as it would if the user had given an input, except that it is done in the simulation environment.

Similarly, when the system wants to evaluate a sentence `believe(`α`,`φ`)`, the specialist is queried. It finds α's simulation environment and temporarily makes it the primary environment, as above, and then calls the same query function that a user would call, but with the effort level turned down (see page 59). If the query answers in the affirmative, then the specialist indicates that `believe(`α`,`φ`)` is true.

Given this mechanism and a sentence of a certain form, it is possible for the system to enter an infinite chain of nested simulations. In such a case, the system may never return when asked a query, and therefore cannot be modeled as a belief machine. This points out the fact that when using our model to evaluate the appropriateness of adding simulative inference to a system, it is the entire system *with simulative inference added* that must be evaluated. For the time being, we are avoiding infinite regress by enforcing a fixed cutoff at three levels of nesting. In the future, heuristics might be used to allow deeper nesting when it is potentially useful.

The global variables which comprise the state of the inference mechanism are given a new dynamic scope during a simulated *ASK* or *TELL*, so that a simulation can take place in the middle of an input-driven or goal-driven proof attempt, as a step of that proof. Note that the hash table containing simulation environments is itself stored in an environment, as part of the belief specialist's state in that environment, so that simulations can be nested to an arbitrary depth.

Being implemented as a specialist module, the simulation mechanism is subject to the limitations of the specialist interface. Among these is the fact that only ground atoms are passed to specialists for evaluation, so simulative inference can't be used to help prove quantified-in belief sentences, or sentences in which the believer term is a

variable of quantification. If this limitation could be overcome, there would be an interesting possibility of reasoning about quantified-in sentences via simulative wh-queries: intuitively, there is an x which John believes to be P if John can answer the question "What is P?". This would involve extensions of both the theoretical framework and the program's interface to simulations.

Another, more serious limitation involves reasoning about changes in an agent's beliefs. EL uses a variant of situation semantics, in which a sentence may be asserted to be not simply true, but true in a particular situation, which may have limited temporal (and/or spatial) scope. Therefore, the knowledge representation can support reasoning about an agent's beliefs at a particular time, and how beliefs change over time. But the specialist interface only passes timeless propositions to the specialists, not assertions about what holds in a particular situation, and so the belief specialist can't contribute to reasoning about belief change. For all of the previous specialists, this was an acceptable limitation: for example, the type specialist needn't reason about change over time because the truth of assertions like "Dogs are mammals" does not vary over time. But agents' beliefs do vary over time, in important ways, and so this limitation will eventually need to be overcome. While this limitation of the specialist interface is the most immediate obstacle to reasoning about belief change, it is not in fact the most difficult one. If the specialist were able to receive information about agents' beliefs at particular times, it is by no means clear what it should do with this information. Some of the problems that arise look quite similar to those we have been addressing: an agent's beliefs tomorrow will be largely similar to his beliefs today, just as one agent's beliefs are largely similar to another, so it would be wasteful to build a separate environment for each time at which one wants to reason about an agent's beliefs. Reasoning about belief change is an interesting problem, and one of quite large scope.

There is also a conceptual liability to using the scheme of reasoning with multiple environments. Before we modified the system, the meaning of sentences stored in EPILOG was clear, since EL has a rigorously defined formal semantics. But now, to interpret a token of a sentence in the system, we must consider not only the sentence itself, but the environment in which it is stored. Actually, this was already true in the system before our changes in some of the specialist representations, since they maintain a separate structure for each propositional attitude context. But the situation is even more complex now, since we have defined environments such that one can be accessible from another. There is no formal statement of, for example, what state of the world is described by an instantiation of the system in which a sentence is present in the common world knowledge environment, and its negation is present in the system's private environment. The reason the practice of AI involves formal semantics is so that systems can be built in a principled way, such that their state can be formally interpreted. By partitioning the knowledge base into multiple environments with different semantics, we have acted somewhat counter to this desire for clear interpretability. This is not to say that EPILOG's states could not be formallly interpreted.

Logics of common knowledge, and logics of context [Dinsmore, 1991; Buvač, 1993; McCarthy, 1993] might be useful in this direction.

4.6 Evaluation

We have made significant changes to EPILOG in order to implement simulative inference, and so we should attempt to demonstrate that these changes don't impact the system's performance, or that they impact it to a small enough degree to be warranted by the increased inferential ability. But evaluating common-sense inference systems is tricky. Such systems are designed to be used with a large knowledge base of common sense facts, but no such knowledge base is currently available. The problem of acquiring, in a computer-usable form, the vast amount of information that all humans know is the most important and most difficult obstacle to the development of logical AI; this dissertation makes no contribution towards its solution.

In the absence of a comprehensive world knowledge base, we must settle for other means of verification. One method is to choose an example question to ask the system, decide what knowledge a human would use to answer it, and enter that knowledge by hand before making the query. This sort of testing demonstrates the kind of inference the system is capable of, but shows little about performance. When using a small, hand-coded knowledge base, one might do quite well using any logically complete theorem prover; but when the knowledge base is much larger, selectivity and the right heuristics become crucial.

To test how the system behaves with a larger knowledge base, we can use randomly-generated "knowledge" in place of genuine world knowledge. This method is no more useful for testing the validity of the designer's assumptions, because the very same assumptions must be used to generate input with what we believe to be realistic statistical properties. But large random datasets can at least test how well the design has been carried out, *i.e.* how well the system can be expected to perform if the assumptions are correct.

Spending a significant amount of time fine-tuning the performance of a system like EPILOG without having the real data with which it is designed to work would be rather futile. For the time being, EPILOG is better viewed as a prototype, an illustration of concepts, than as a practical system. Therefore, we are content with rough estimates of the impact of our changes. Our modifications to the general-purpose reasoning mechanism, and to the set and equality specialists, are all of the same character: where once a single hash table lookup was performed, now two are performed, one in the primary environment and one in the secondary. From this description alone, we can't predict by exactly what fraction the system's run time will increase for an average query, but we are convinced that it will be rather small. Therefore, we have not carried out extensive performance testing of the system as a whole. We will simply present, in Section 4.6.2,

a transcript of the system processing a sample story, as a demonstration of the type of inference it can perform. The changes we made to the time specialist are potentially of more significant impact on performance, since they involved replacing direct pointer references with hash table lookups, in a graph structure that is potentially quite large. Therefore, we have written code to generate large amounts of temporal information of the sort processed by the specialist, enter the information into the specialist, and make queries, and we have collected timing data both for the version of EPILOG from before our changes and from our modified version, in order to measure the impact of our changes.

4.6.1 Performance Study of the Time Specialist

The time specialist implements the TimeGraph I algorithm for temporal reasoning [Gerevini *et al.*, 1995; Miller and Schubert, 1990, see also the *Epilog User's Manual*, available along with the source code at the URL given at the beginning of this chapter.]. This algorithm is specifically designed to work efficiently with the kinds of temporal information that narratives typically convey: sequences of events for which a total ordering is known, with less information about relative ordering between events in different sequences. A node of the graph represents an instantaneous point in time, and a link represents a before/after relationship, possibly with a duration, between two points. An event is represented by a pair of points, the start point and the end point. A totally ordered sequence of points is called a *chain*. A time point can belong to only one chain—even if it belongs to more than one totally ordered sequence, it is placed in only one chain, and connected to the other sequence(s) by cross-chain links. The points in a chain are marked with their number in the sequence, so that a before/after determination between any two points in a chain can be made in constant time, by comparing their indices. Queries about the relative order of two points on different chains are answered by searching the graph for a path between the two; a "meta-graph," which stores information about links between chains, and the labels on nodes in each chain speed up the search.

A node may be marked with an upper bound, a lower bound, both, or neither, on its absolute time (measured in seconds since a fixed zero point). An edge may be marked with an upper bound, a lower bound, both, or neither on its duration.

Our graph generation code creates random graphs with specified statistical properties. In the first step, it chooses a set of points on a bounded interval, with the following parameters (in the test whose results are reported below, we varied the number of chains from five to 560, and set the other parameters to the values in parentheses):

- the number of chains
- the expected length of each chain, chosen with a geometric distribution (20)

- the expected ratio of the duration of a chain to the duration between the maximum and minimum time. Chains are generated one at a time, and a running average of this ratio is kept; for each new chain, if the average for the previous chains is above the expected value, the new chain's ratio is chosen between zero and the expected value with a uniform distribution, and if the average was below the expected value, it is chosen between the expected value and the maximum with a uniform distribution. (0.3)

In a second step, the code generates an imprecise description of the set of points from step one, by choosing upper and lower bounds that surround each absolute time and the duration of each interval. This process has the following parameters, again with their values for the reported test suite in parentheses:

- the expected ratio of transitive edges (edges from one node to another in the same chain which is not its immediate successor) to direct edges, chosen for each chain with a geometric distribution (0.3)
- the ratio of cross-chain edges to direct in-chain edges (0.3)
- uncertainty: the expected size of the interval between a point's upper bound and its lower bound, as a fraction of the durations of the links on either side; also used as the expected size of the interval between the upper and lower bounds on a link's duration, as a fraction of that link's duration (0.1)
- the probability that a point's upper bound will be reported to EPILOG; also the (independent) probability that its lower bound will be reported (0.1)
- the probability of a link's duration being given a non-zero lower bound (all durations are given upper bounds) (0.5)

Finally, a description of the resulting information is generated in EL, using the following types of sentences, where T1 represents a time point name, T2 represents either a time point name or an absolute time, and D represents an absolute duration:

```
(T1 before T2)
(T1 after T2)
(T1 at-most-before T2 D)
(T1 at-least-before T2 D)
```

We used this code to generate a test suite of graphs with nineteen different graph sizes, and three graphs at each size. We also generated 1,000 random queries of the form `(T1 before T2)`, where T1 and T2 are both time point names, for each graph. We entered each graph description into both the original version of EPILOG and our modified version, and posed all of the queries to each, and recorded the time taken to

construct the graph and to answer the queries. The results are shown in Figures 4.7 and 4.8. They show that the overhead introduced by our modifications increases graph construction time negligibly for graphs of less than 4,000 nodes, and by about a third for very large graphs of 13,000 nodes. Query times were not significantly affected. We cannot account for the sharp drop in complexity for graphs around 10,000 nodes.

Figure 4.7: Time graph construction times

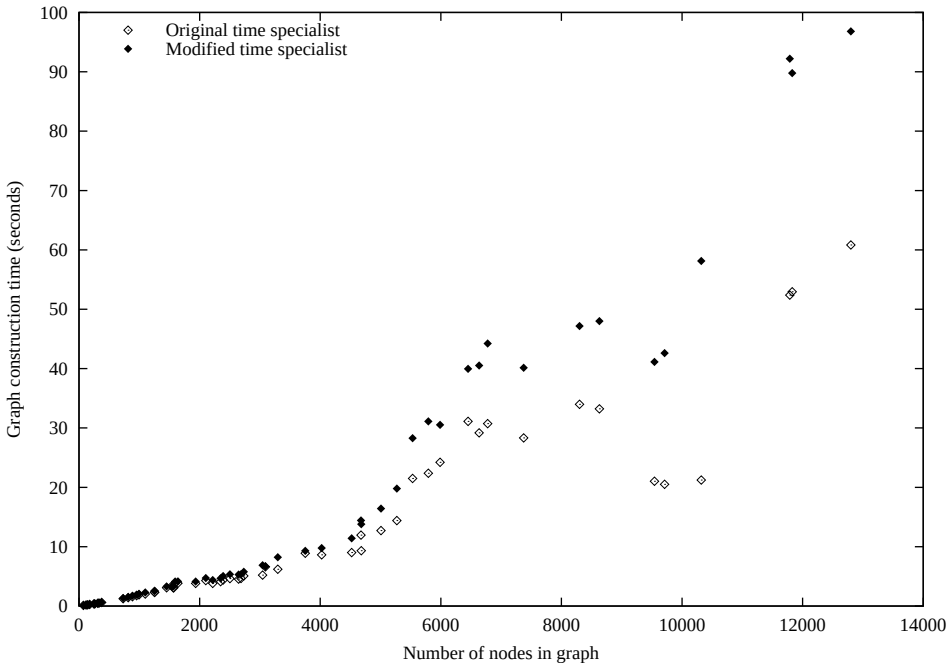
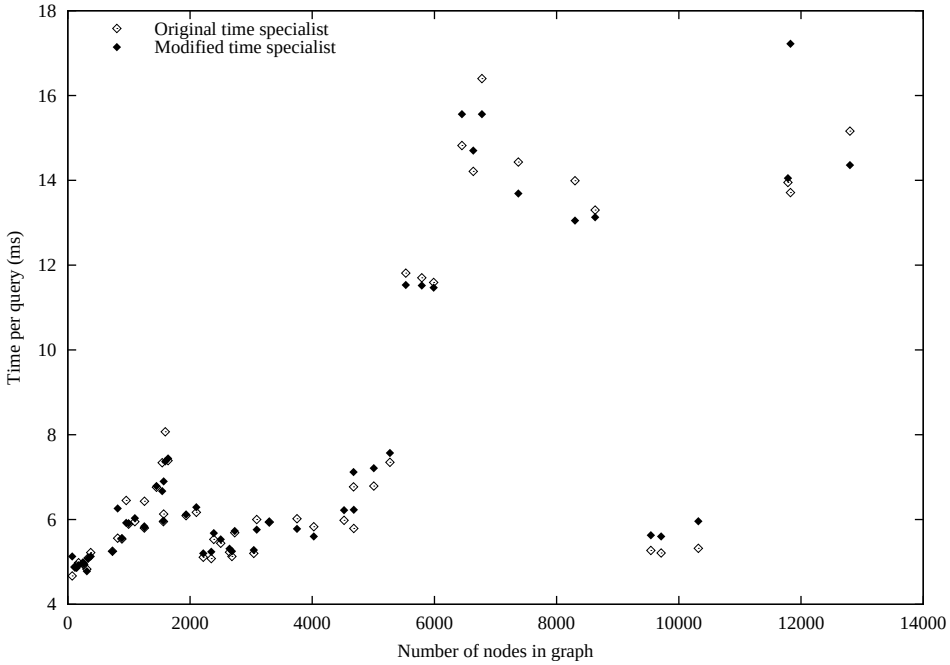


Figure 4.8: Time graph query times



4.6.2 An Example Story

Over the years of EPILOG's development, the story of Little Red Riding Hood has been used as an ongoing test and example. We will now demonstrate that simulative inference brings EPILOG closer to understanding one fragment of the story.

The sentence to be interpreted is the following: "The wolf would have liked to eat Little Red Riding Hood, but he dared not on account of some woodcutters nearby." The challenge is to explain the connection between the woodcutters' presence and the wolf's decision not to eat Little Red Riding Hood (henceforth LRRH). Human readers understand that the wolf believed that if he tried to eat LRRH, the woodcutters would do something about it. We would like EPILOG to make the same inference.

The example below consists of input given to EPILOG. The information is expressed in a LISP-friendly version of Episodic Logic. See the references listed at the beginning of this chapter for the details of EL; here are a few important points for understanding the example:

- Each expression is a LISP list (surrounded by parentheses).
- For similarity to English syntax, a predicate follows its first argument, *e.g.* "LRRH is a girl" is represented as `(LRRH girl)`.

- Episodic Logic takes its name from its use of episodes, which are similar to the situations of situation semantics. They are elements of the domain of discourse, can be referred to by terms, and may have temporal and/or spatial extent. The operator `**`, an infix operator, relates a sentence to an episode that the sentence characterizes. For example, the sentence `((John run) ** e1)` means that e1 is an episode of John running.

- The syntax of quantification is richer than that of ordinary first-order logic. A quantified formula consists of a quantifier, an optional restrictor, and the matrix. This allows quantifiers such as "most" which aren't easily expressed in ordinary FOL, but for the universal and existential quantifiers is only a syntactic difference.

  ```
  (A x (x man) (x mortal))
  ```

  is equivalent to

  ```
  (A x ((x man) implies (x mortal))),
  ```

  and

  ```
  (E x (door x) (green x))
  ```

  is equivalent to

  ```
  (E x ((door x) and (green x))).
  ```

- The operator `Ka` forms a term from a predicate. `(Ka run)` is a kind of action, namely running.

- The operator `pair` takes terms denoting an individual and an episode, and forms a term denoting the action performed by that individual in that episode.

- The operator `that` takes a formula and makes a term denoting a proposition.

- The operator `coll` takes a predicate and forms another predicate. `(x (coll woodcutter))` means that x is a collection of woodcutters.

- An underscore and a label after a term indicate the term's sort. The label only needs to be used on the first occurrence of the term; the system remembers it after that. The label `_ep` indicates a term that denotes an episode.

- The quasi-quotation operator `qq` allows one to give EPILOG rules of inference that can't be expressed as formulas of EL. It creates an environment in which both ordinary EL expressions and variables over EL expressions are permitted. When variables over EL expressions appear in a `qq` environment, and are bound by a quantifiers outside of that environment, the quoted expression's meaning is the result of replacing the variables with their values.

In EPILOG, the functions `meta`, `mp` (for “meaning postulate”), `kn` (“knowledge”), and `story` are all ways of entering information, and the function `q` is for asking questions.

The global variable `*environment*` indicates the current primary environment. It normally points to the system’s private environment, but it is automatically set to a simulation environment during a simulation, and the user can set it manually. In the example, the forms within the scope of the

```
(let ((*environment* *wkb*)) ...)
```

have their effect on the environment that stores the shared world knowledge base.

We will now display the input that the system processes, and then discuss some of the representational and inferential issues involved. The following world knowledge is entered into the shared environment:

```
;; Everything within this 'let' is common knowledge, i.e. believed by
;; everyone.
(let ((*environment* *wkb*))

  ;; Eating, attacking, and punishing are actions.
  (meta
   '(eat %action-pred)
   '(attack %action-pred)
   '(punish %action-pred)
   )

  (mp

   ;; If there is a collection of things of some type, then there is a
   ;; thing of that type which belongs to the collection.
   '(A p_pred (A y_set ((qq (y (coll p))) true)
               ((qq (E z (z member-of y) (z p))) true)))


   ;; Rules to convert between the equivalent forms
   ;; ((x pred) ** e) and ((pair x e) instance-of (Ka pred))

   '(A p_pred (p %action-pred)
     (A e1_term (A x_term ((qq ((pair x e1) instance-
of (Ka p))) true)
                  ((qq ((x p) ** e1)) true))))
   '(A p_pred (p %action-pred)
     (A e1_term (A x_term ((qq (not (pair x e1) instance-
of (Ka p))) true)
                  ((qq (not (x p) ** e1)) true))))
   '(A p_pred (p %action-pred)
     (A e1_term (A x_term ((qq ((x p) ** e1)) true)
                  ((qq ((pair x e1) instance-of (Ka p))) true))))
   '(A p_pred (p %action-pred)
```

```
   (A e1_term (A x_term ((qq (not (x p) ** e1)) true)
               ((qq (not (pair x e1) instance-of (Ka p))) true))))
 )

(kn

 ;; If a creature believes that an action will have undesirable
 ;; consequences, he won't perform that action.
 '(A x (x creature)
   (A y (y action-type)
    (A context-ep_ep
     ((x believe
         (that
          (A action-ep_ep
             (((pair me action-ep) instance-of y)
              and
              (action-ep during context-ep))
             (E result-ep_ep (action-ep cause-of result-ep)
                (result-ep undesirable)))))
      implies
      (A e1_ep (e1 during context-ep)
         (not (pair x e1) instance-of y))))))

 ;; I am a creature
 '(me creature)

 ;; Being punished is undesirable.
 '(A e1_ep
   (A x ((x punish me) ** e1)
    (e1 undesirable)))

 ;; Eating a living creature involves attacking it
 '(A x (x creature)
   (A y ((y alive) and (y creature))
    (A e1_ep ((x eat y) ** e1)
     (E e2_ep (e2 during e1)
      ((x attack y) ** e2)))))

 ;; To attack a child is wicked
 '(A x (x creature)
   (A y (y child)
    (A e1_ep ((x attack y) ** e1)
     ((pair x e1) wicked))))

 ;; Doing something wicked when someone is nearby will bring
 ;; punishment.
 '(A x (x creature)
   (A y (y human)
    (A e1_ep ((pair x e1) wicked)
     (A e2_ep ((y near x) ** e2)
      (E p
```

```
            (E e3_ep (e1 cause-of e3)
               ((p punish x) ** e3)))))))

   ;; Woodcutters are human
   '(A x (x woodcutter) (x human))

   )
  )
;; End of the common knowledge
```

In addition to the above knowledge, the system has a taxonomic predicate hierarchy, from which it knows, among other things, that girls are humans, and that wolves and humans are creatures.

Using this knowledge, the system can process the following story fragment:

```
(story

 ;;LRRH met a wolf.
 '(E meet-ep_ep
   (E wolf1 (wolf1 wolf)
    ((LRRH meet wolf1) ** meet-ep)))

 ;; Little Red Riding Hood is a girl, and the wolf knew that.
 '(LRRH girl)
 '(wolf1 believe (that (LRRH girl)))

 ;; She was alive, and the wolf knew that.
 '(LRRH alive)
 '(wolf1 believe (that (LRRH alive)))

 ;; There were woodcutters nearby, and the wolf knew that.
 '(E y_set (y (coll woodcutter))
   (A x (x member-of y)
    (E se3_ep (meet-ep during se3) ((x near wolf) ** se3))))
 '(wolf1 believe
   (that
    (E y_set (y (coll woodcutter))
     (A x (x member-of y)
      (E se3_ep (meet-ep during se3) ((x near me) ** se3))))))

 )
```

Given all of this information, the system answers “no” to the question “Did the wolf eat Little Red Riding Hood when they met in the forest?”, asked as follows:

```
(q '(E e1_ep (e1 during meet-ep)
     ((wolf1 (eat LRRH)) ** e1)))
```

The inference involved in answering the question includes a simulative inference step, in which it is derived that the wolf believes that if he eats LRRH during the episode of the story fragment, then an undesirable episode will result. The sentence that the system proves that the wolf believes is

```
(A EP-X (((PAIR ME EP-X) INSTANCE-OF (KA (L L-U (L-U ATTACK LRRH))))
        AND
        (EP-X DURING VAR18))
   (E EP-V (EP-X CAUSE-OF EP-V) (EP-V UNDESIRABLE)))
```

This question is not exactly the one we had hoped the system could answer, although the simulative inference involved is the same. The natural translation into EL of the question "Why didn't the wolf eat LRRH" says roughly "What episode is the cause of the episode of his not eating her?", and the answer is the episode of his believing that it would have undesirable consequences. As we mentioned in Section 4.5, the belief specialist currently only reasons about belief as an eternal state, and can't associate beliefs with episodes. Consequently, the system can't prove that there is an episode of the wolf believing something.

An important caveat about the example is that the effort level used in simulations (see p. 59) is a tunable parameter, and the system can only answer the question when all of its inferential abilities are enabled in simulations. As we discussed in Section 4.2, this causes the belief machine to violate the closure constraint, and therefore raises the possibility that simulative inference will generate incorrect conclusions.

We provided the system with the information that the wolf knew LRRH was a girl and alive. These facts are of course not given explicitly in the original story—the system should infer them from the fact that the wolf could see LRRH, and that the properties of being a girl and alive are readily apparent. There is no reason in principle that EPILOG couldn't do this.

Little world knowledge is certain. Many of the rules we have given the system would be more appropriately expressed as probabilistic rules, so that it could use them defeasibly. The form of simulative inference we use is not sound in a system that reasons defeasibly, but a more sophisticated form, in which the system reasons explicitly about beliefs the agent doesn't have, could be both appropriate and useful.

EPILOG doesn't have facilities for planning and action, but if it did, it might process the story in a more natural way. We gave the system the knowledge that creatures avoid actions they believe will have undesirable consequences, and that being punished is undesirable, in a declarative form; but an artificial agent designed to act in the world would have that information encoded procedurally, and it might very naturally attribute such dispositions to others by simulation, just as we have done for belief.

Much work remains to be done, to improve the efficiency of simulative inference and to increase the range of situations in which it can be applied effectively, but even the current state of the implementation is a useful extension of EPILOG's original capabilities.

5 Other Work on Reasoning About Belief

In Section 2.5.1, we gave a brief overview of one branch of literature on belief. That community is composed primarily of philosophers of language, and their disputes tend to be about whether a given theory of meaning is sufficiently fine-grained, *i.e.* whether or not English sentences which according to the theory have the same meaning can in fact differ in truth value. The work described in this dissertation belongs to a different line of inquiry, one followed primarily by researchers in AI. For this community, the primary emphasis is on inference. We are concerned with what can be inferred about someone's beliefs from more directly observable phenomena and from (implicit or explicit) knowledge about how belief systems function, and with what can be predicted about someone's actions based on knowledge about his beliefs. In this chapter, we explore some of the literature on reasoning about knowledge and belief. After giving a survey of this literature, we will return in more depth to one model, Konolige's deduction model of belief, which has important similarities to ours. See [Fagin *et al.*, 1995] for a more extensive survey of the field.

5.1 Possible Worlds Theories

The point of departure for much work on reasoning about belief is the possible worlds model of Hintikka [1962], with important refinements by Kripke [1963]. According to this elegant theory, if one knows that α believes φ, and φ entails ψ, then one can infer that α also believes ψ. This can often be a useful sort of inference for predicting behavior, but it is an idealization, not a realistic model of human belief. Entailment in any logic at least as expressive as first-order logic is undecidable, so according to the model agents can have uncomputable belief sets. This offends the sensibilities of proponents of AI, whose fundamental hypothesis is that cognition is computation; but one needn't rely on that identity to show that the possible worlds model doesn't describe human belief. Many people know the rules of chess, yet no one knows whether white can force a win from the beginning of a game, even though the answer to that question is determined by the rules.

Not all authors in computer science have rejected the classical model, despite the property of logical omniscience; in fact, the model has been built upon and extended for many different purposes. Like the frictionless surface of physics problems, logical omniscience is an idealization which can be useful as a means of breaking down a complex phenomenon into more easily analyzable parts. However, in this dissertation we have explicitly set out to define a model in which logical omniscience does not obtain, so we will leave aside that branch of the literature.

Several other models have used the ideas of the possible worlds model, but modified in such a way as to eliminate logical omniscience. In the classical model, a possible world is essentially an ordinary first-order model. Levesque's influential model of "explicit belief" [1984], in contrast, uses *situations* which are models of a four-valued logic: a situation can support the truth or falsity of an atomic sentence, or both, or neither. The truth and falsity of complex sentences are determined recursively by rules for each of the logical connectives; notably, the rule for negation is that a situation supports the truth of $\neg\varphi$ iff it supports the falsity of φ, and *vice versa*. Consequently, there can exist a situation that supports the truth of φ and of $\varphi \supset \psi$, but not of ψ: a situation that supports the falsity of φ supports the truth of $\neg\varphi$, and therefore of $\neg\varphi \vee \psi$, or equivalently $\varphi \supset \psi$; such a situation can also independently support the truth of φ; and we have placed no constraints at all on the truth or falsity of ψ. Because of the existence of such situations, the theory can model an agent who believes φ and $\varphi \supset \psi$ but not ψ. As in the classical possible worlds model, belief in $\varphi \wedge \psi$ still entails belief in φ, and belief in φ entails belief in $\varphi \vee \psi$. Levesque's original logic was a propositional one, but it has been extended to a first-order logic in [Lakemeyer, 1994] (in addition to blocking embedded *modus ponens*, the propositional version blocks embedded existential generalization from disjunctions, *i.e.* believing $P(a) \vee P(b)$ doesn't entail believing $\exists x P(x)$); the original proposal didn't allow for nested belief, but this is solved in [Delgrande, 1995].

In Levesque's model, believers can be seen as perfect reasoners in a weaker-than-ordinary logic which is decidable, and in fact tractable (we have paid little attention to the question of tractability in this work, but it was one of the explicit desiderata for Levesque). Since the logic is decidable, there is a belief machine that implements a sound and complete inference procedure for it, and so there is an instantiation of our logic that is equivalent to Levesque's. Furthermore, there are other instantiations of our logic that are more interesting than Levesque's—ones in which the belief machine is less minimal, more like human belief. Some authors, for apparently aesthetic reasons, prefer "purely semantic" models like Levesque's to ones like ours in which syntactic entities (sentences of a language of belief) are part of the semantics. But to us, it seems unlikely that belief can be fully explained or described without making reference to the mental representations of the believer, and so it seems entirely appropriate for a model of belief to include representations.

Levesque's model of belief has been used as a tool in some work on subjects related to introspection and nonmonotonic reasoning, *e.g.* [Levesque, 1990; Lakemeyer, 1997].

It would be interesting to explore the possibility of using another logic of belief in the same role.

Fagin and Halpern [1988] suggest a version of possible worlds semantics in which an agent may have many frames of mind. If an agent believes both φ and ψ, but in different frames of mind, then he believes all of φ's consequences and all of ψ's consequences, but not necessarily any consequences of their combination. The entailments concerning belief are therefore even weaker than those in Levesque's model. Fagin and Halpern cite [Stalnaker, 1984] and an unpublished manuscript of Levesque as earlier statements of the idea of multiple frames of mind. Similar ideas are also used in [Lenat and Guha, 1990] and [Giunchiglia *et al.*, 1993].

5.2 Sentential Theories

We now turn to models which, like ours, define belief in terms of sentences, and use the concept of inference in the definition of belief. None of these models makes the assumption of logical omniscience.

Our logic can be viewed as a refinement of that of Eberle [1974]. That logic has an inferability predicate I, where $I(\varphi_1, \ldots, \varphi_n; \psi)$ means that ψ is inferable from $\varphi_1, \ldots, \varphi_n$. The inferability relation can be an arbitrary set of tuples of sentences, so logical omniscience needn't obtain. An axiom of the logic says that if $\varphi_1, \ldots, \varphi_n$ are believed, and ψ is inferable from $\varphi_1, \ldots, \varphi_n$, then ψ is believed. This is clearly analogous to our rule of simulative inference. Eberle also shows that if a constraint analogous to our closure constraint is placed on the inferability relation, then his logic is sound and complete. Our model extends that of Eberle by treating the premises of simulative inference as a sequence rather than a set, and by defining the inferability relation in computational terms.

Halpern, Moses, and Vardi [1994] describe a model of "algorithmic knowledge" that is similar to our model in that an agent is modeled has having a reasoning algorithm, and its beliefs are taken to be the sentences that the algorithm can verify given the information encoded in the agent's knowledge state. The agent's reasoning algorithm is analogous to our *ASK* function, but their model contains nothing analogous to our *TELL* function, *i.e.* the model doesn't specify the process by which an agent comes to be in a particular knowledge state, or how learning new information would cause the state to change. The technique of simulative reasoning has not been discussed in the framework of algorithmic knowledge.

The mode of reasoning described by Haas [1986; 1995] is also simulative reasoning, but of a different kind. It can be summarized as follows: if α believes φ at time t, and I can prove ψ from φ in n time units, then α believes ψ by time $t + n$. Since beliefs are indexed with a time at which they are known to be believed, this simulative technique is sound for a broader class of reasoning mechanisms than ours. Specifically, the closure

constraint is not required for soundness, because the logic differentiates between base beliefs and derived beliefs. However, implicit in Haas' presentation is the assumption that the computation that generates new beliefs is undirected forward inference, simply the enumeration of conclusions that can be reached from a set of premises, not goal-directed inference that attempts to verify or refute a particular query sentence. If the mechanism being used in simulative reasoning were a goal-directed theorem prover, then one would not be justified in concluding that another agent with similar inferential ability had reached the same conclusion in the same amount of time. For simulation to be justified, one would have to know, in addition, that the agent had had occasion to "wonder" about the query sentence, *i.e.* had begun attempting to prove it. This kind of information seems unlikely to be available in many ordinary situations.

Related ideas of time-bounded inference can be found in the "step logics" of Elgot-Drapkin and Perlis [1990]. In these logics, an agent's beliefs are seen as a set of sentences that changes over time, at integral time steps, as a result of both observation and inference. Inference can introduce new sentences derived (via sound or unsound rules) from sentences believed in the previous time step, and can also cause the retraction of beliefs held in the previous step, as a result of contradictions discovered in the earlier belief set. This nonmonotonicity makes the formalism more general than that of Haas, but by the same token makes it less likely to be useful for reasoning about the beliefs of others: Haas' method of reasoning about belief requires only that one have an upper bound on the time at which an agent learned a particular fact, but a step logic requires that one know the *precise* time at which each relevant fact was learned, since a conclusion could be drawn from that fact early on, but later retracted. Indeed, Elgot-Drapkin and Perlis make no claims about simulative inference. The step logics come in pairs, consisting of an agent language which a machine could use for maintaining its beliefs, and a meta-language characterized as a "scientific theory" of such a machine's beliefs, for use by humans in thinking about the machine's behavior, but not intended as a knowledge representation language for use in a computer program.

The logics of [Haas, 1995], [Perlis, 1988], and [Grove, 1995] have in common that they allow explicit quantification over names. In our model, in contrast, while quantifying-in is implicitly quantification over names, terms of the logic are not elements of the domain of discourse. As a result, our logic is not quite as expressive; Grove gives some examples of situations in which reasoning explicitly about names is desirable. The tradeoff is that both the syntax and the inference methods for our logic are significantly simpler.

Finally, the deduction model of belief of Konolige [1986] is similar to ours in important ways. We devote Section 5.5 below to a comparison of the two.

5.3 Implementations of Simulative Inference

Our work, like that of Konolige and Haas, provides both a technique for reasoning about belief by simulative inference, and a formal definition of belief involving inference. Many more systems have used the idea of simulative inference, but without the semantic basis of an inferential model of belief.

In [Moore, 1973], Moore sketches an imagined system that would represent the proposition "α believes φ" by creating a separate database for α's beliefs, and storing φ in it. Later queries of the form "does α believe ψ?" would be answered by evaluating the query ψ in the context of that database. Moore later rejected this scheme [1977], pointing out that such a system would be unable to represent disjunctions of belief sentences (*e.g.* "either α believes φ, or he believes ψ") and negations (*e.g.* "α doesn't believe φ"). His solution in [Moore, 1977] was to abandon simulative reasoning for a technique that involves reasoning explicitly about possible worlds.

Creary [Creary, 1979] pointed out that the faults Moore found with the database approach were valid arguments against using reasoning contexts for the *storage* of information about beliefs, but were not reasons to abandon simulative reasoning as an inferential technique. Creary envisioned that facts about belief would be stored in their objective form, a form that makes use of a belief operator, but that when simulative reasoning became appropriate, a new reasoning context could be created and the contents of any relevant beliefs entered into it, just prior to a simulative query being made.

Unfortunately, as we discussed in Section 4.3, in real applications a system is likely to have an enormous amount of information about what other agents believe, and identifying the beliefs that are relevant to a particular query is by no means a trivial problem. Therefore, in EPILOG we have chosen to use the "database approach," *i.e.* maintaining a persistent reasoning context for each agent about whose beliefs the system has some information, as much as possible, while also having a belief operator in the general-purpose representation language so that disjunctions and negations of belief sentences can be stored in objective form. (We must note that while EPILOG's knowledge representation is adequate for storing disjunctions and negations of beliefs, it remains to be seen whether the inference mechanism can make use of this information fruitfully and efficiently.)

The main assertion of the work of Chalupsky and Shapiro [Chalupsky and Shapiro, 1996; Chalupsky, 1996] is that one can never know for certain what inferences someone else will make, and that therefore conclusions reached by simulative reasoning must be defeasible. They have built an inference system that keeps track of the hypotheses and assumptions upon which each derived sentence depends, so that if a hypothesis is withdrawn, or an assumption proves to be incorrect, the conclusion will no longer be believed. A conclusion drawn by simulative inference, although it is a derived conclusion, is also marked as an assumption, and if it is contradicted by more authoritative

information (*i.e.* information not derived by simulative inference), it ceases to be believed.

Incidentally, Chalupsky and Shapiro eschew truth-conditional semantics, but they define a justification relationship which is intended to be analogous to the notion of entailment in ordinary logics. Given that one has some set of base beliefs, this relationship defines mathematically the set of propositions that one is justified in concluding. The relationship is defined so that if someone with base beliefs $\varphi_1, \ldots, \varphi_n$ is justified in concluding ψ, then someone with base beliefs $B(\alpha, \varphi_1), \ldots, B(\alpha, \varphi_n)$ is justified in concluding $B(\alpha, \psi)$. In other words, simulative inference is always *justified*, but sometimes it can be *incorrect*—sometimes an agent may simply fail to make an inference he might well have made. (This is surely the only time an author has claimed an inference rule to be both sound and defeasible [Chalupsky, 1996, p. 164].)

Given that current machines are very far from imitating human reasoning, and that even humans can predict the beliefs of other humans only in very controlled circumstances, the idea of making simulative inference a defeasible rule is a very useful one, and Chalupsky and Shapiro have demonstrated in great detail how the techniques of default logic and assumption-based truth maintenance can be applied to reasoning about belief. We acknowledge that in practice the conditions for sound simulative inference are rarely met, but nevertheless it is important to identify these conditions, in order to be able to recognize the various ways they can be violated. Treating an inference rule as defeasible only prevents it from swamping the system with absurdities if the rule's conclusion is contradicted by other information. Not all incorrect inferences are immediately contradicted, and defeasibility doesn't protect the system from making poor decisions based on incorrect assumptions. Ultimately, simulative inference should be a defeasible rule, but one which is only applied in situations where it might be correct.

In Section 4.4, we noted a similarity in the representational needs of simulative inference and assumption-based truth maintenance. Creary alluded to this similarity when he suggested using a CONNIVER-like system of reasoning contexts for simulative reasoning, and the similarity is realized in both SIMBA [Chalupsky, 1996] and Rhet [Allen and Miller, 1991], each of which handles hypothetical reasoning and reasoning about belief with a single mechanism. SIMBA's justification management system is simply to keep a list of the hypotheses on which each conclusion depends. As we pointed out in Section 4.4, while this technique is attractive for its simplicity, it can't be used in EPILOG because of the non-sentential knowledge representations used by the specialists, which can conflate the contents of multiple hypotheses, so that it becomes impossible to obtain an exact list of the hypotheses used in a proof. Our alternative was to use environments that combine all of the "hypotheses" that a particular agent is thought to believe. This is also the approach used in Rhet, and for similar reasons, since that system has an equality specialist like that of EPILOG.

5.4 Related Issues

Our work on reasoning about belief has focused on simulative inference. This technique can only be applied when one has already attributed some initial beliefs to an agent, and we have said relatively little about how to obtain those premises. Others have written about this question. Isozaki [1995] writes about ascribing belief by reasoning about the observability of actions and common knowledge of their consequences. Jameson [1995] discusses (probabilistically) ascribing beliefs to a person based on, for example, the ascriber's own beliefs, and knowledge about typical beliefs for some group. Ballim and Wilks [1991] also write about a reasoner that ascribes its own beliefs to others, using a default framework rather than a probabilistic one. Interestingly, their point of view seems to be that any logicist theory of belief must try to characterize belief sets using only semantic properties. They rightly hold that this can only lead to idealized, unrealistic models, because it ignores the computational mechanism that generates the belief sets; they cite this as justification for using a less principled approach to reasoning about belief. Our work incorporates computational mechanisms into reasoning about belief, without abandoning the logicist paradigm.

5.5 Konolige's Deduction Model

In its essence, our model of belief bears a strong resemblance to Konolige's deduction model [1986]. In the deduction model, a believer is represented by a *deduction structure*, which is composed of a set of sentences (the base beliefs) and a set of inference rules. An agent's belief set is the set of all sentences that can be derived from the base beliefs by exhaustive application of the inference rules. Agents that are not logically omniscient can exist in the deduction model, because the set of inference rules is not required to be complete.

The fundamental difference between the deduction model and ours is the contrast between deduction structures and belief machines, and we will focus primarily on the consequences of this difference. However, at the end of this section we will briefly mention some other differences.

5.5.1 Expressiveness

Neither model is strictly more expressive than the other. That is, each model is capable of describing certain agents that can't exist in the other. But, as we will now show, the agents that exist only in the deduction model are merely mathematical constructions that have no implementation as actual programs, while among those that exist only in the computational model are some that reason in interesting and tractable ways.

One thing that distinguishes the computational model from the deduction model is that it describes not only agents' beliefs in a single state, but how their beliefs change when they learn new information. In the deduction model, learning something new can only be modeled as adding a new sentence to the base beliefs. This is sufficient when the new information is consistent with the agent's previous beliefs, but when it isn't, it means that the agent must believe a contradiction. Naturally, to restore consistency the agent should retract some of its previous beliefs, but a deduction structure cannot describe how an agent chooses which beliefs to retract. A belief machine, in contrast, can encode this kind of information. The function *TELL* describes how the belief machine's state changes in response to any sentence, not only ones that are consistent with what has preceded them.

Because our model of belief can describe reasoners whose beliefs change in interesting ways, we anticipate that it will prove more fruitful to extend to a temporal logic than the deduction model would. But even if we consider only the static logic presented here, the possibility of belief revision distinguishes the computational model from the deduction model. If a belief machine can reject beliefs when it discovers a contradiction, then there may be certain sets of sentences that an agent can never believe simultaneously. For example, it is possible that, upon being *TELL*ed the sentence $\neg P(c)$, a machine might cease to believe $P(c)$, even if it had previously been *TELL*ed $P(c)$. For such a machine, the sentence $B(a, P(c)) \wedge B(a, \neg P(c))$ is unsatisfiable (and also inconsistent, thanks to the negative simulative inference rule). In Konolige's logic, regardless of the set of inference rules used by the agent's deduction structure, the analogous sentence is satisfiable—in the deduction model any base belief set is possible, even an explicitly self-contradictory one.

Negative introspection is one particular type of nonmonotonic reasoning that is possible in our model but not in Konolige's. It is simple to design a belief machine with negative introspection, but there is no equivalent deduction structure because there is no way to write a deduction rule whose premise is the *absence* of a belief from the belief set. Konolige does write a chapter on introspection [ch. 5]; it is about machines that can query themselves, but can *otherwise* be modeled as deduction structures. His use of the negative introspection axiom to describe a class of machines [Theorem 5.4, p. 78] is therefore misleading. In presenting this axiom, he is using the notation of his logic of belief to describe an agent that cannot, in fact, exist in that logic.

For the purpose of further comparison, let us set aside agents that sometimes reject what they are *TELL*ed, and explore how the remaining agents in the computational model compare with agents in the deduction model. Formally, let us add a new constraint, the "credulity constraint," which requires that every sentence is monotonically acceptable in every state, *i.e.*

$$\mathcal{B} \cdot S \cup \{\varphi\} \quad \subseteq \quad \mathcal{B} \cdot \mathit{TELL}(S, \varphi)$$

for every state S and sentence φ. Belief machines satisfying this constraint never reject information or revise their beliefs—they always believe everything they have been

*TELL*ed, even if it is contradictory. With this constraint, the analogy between the two models becomes much closer. For a machine satisfying the closure, commutativity, and credulity constraints, the order of a sequence is never significant (the commutativity constraint applies to all sequences, since since the credulity constraint says that every sequence is acceptable), nor is repetition of elements in the sequence, so to determine an agent's belief set it is sufficient to know the *set* of sentences it has *TELL*ed to its belief machine, regardless of the sequence in which they were presented. The set of *TELL*ed sentences therefore becomes analogous to the base belief set of a deduction structure. The one difference is that the sequence of *TELL*ed sentences must be finite in length, while the base belief set of a deduction structure may be infinite. Any expressiveness gained from this difference is ill-gotten, because no agent that has existed for only a finite amount of time can have explicitly stored infinitely many facts. We will henceforth consider only finite base belief sets.

Under these three constraints, a believer in the computational model can be thought of as a function from finite base belief sets to (possibly infinite) belief sets, as can a believer in the deduction model. But neither model admits all such functions, so we can now compare the two by comparing the set of such functions that each allows.

One readily apparent difference is that in the computational model, belief sets are always computable, while in the deduction model they need not be. For example, if a deduction structure includes a logically complete set of inference rules, then it generates uncomputable belief sets, since the question of logical consequence is only semidecidable. Of course, it is by explicit stipulation that we require *ASK* and *TELL* to be recursive functions, since the possibility of an uncomputable belief set conflicts with our intuitions about what it means to believe.

Further comparison requires that we look more closely at the specification of what sorts of inference rules may be used in a deduction structure. Konolige gives the following two requirements [p. 21]:

Provinciality: The number of input sentences (premises) of the rule is fixed and finite.

Effectiveness: The rule is an effectively computable function of its premises.

As we will show presently, these restrictions are stronger than is actually necessary for Konolige's purposes; but first let us consider how the two models compare if the restrictions are taken literally. Note first of all that the requirement of provinciality, as stated, would seem to imply that rule sets must be finite, since infinite rule sets would allow this constraint to be circumvented: a single rule with a variable number of premises could be replaced by an infinite set of rules, each with a fixed number of premises. Theorem 16 shows that there is a belief machine satisfying all of the constraints we have accumulated so far for which no equivalent deduction structure exists, if the requirements of provinciality and effectiveness are taken at face value.

Theorem 16 *There exists a belief machine m, satisfying constraints C1–C5 and the credulity constraint, for which there is no finite set of provincial, effective inference rules R such that m and R always yield the same belief set when given the same base beliefs.*

Proof: [1] The construction for this proof can be summarized as follows: given a mapping between finite sets of sentences and Turing machines, we construct a belief machine whose belief set is finite exactly when the set of sentences it has been *TELL*ed corresponds (via the given mapping) to a Turing machine that never halts on input 0. This belief machine cannot be modeled by a finite set of deduction rules, because if it could, then the set of Turing machines that never halt on input 0 would be recursively enumerable: a Turing machine would run forever only if the closure of the corresponding set of sentences under the assumed inference rules were finite, and that condition can be effectively detected.

We construct a belief machine so that when it is *TELL*ed a sentence, it simply adds the sentence to a set, which we will henceforth call the machine's *database*. The database is empty in the initial state S_0. The treatment of the sentences as a set will be guaranteed by the definition of *ASK*, which will be order-independent. We can thus think of a database like $\{\varphi_1, \ldots, \varphi_n\}$ as a name for a belief state of the belief machine. Different sets may turn out to name the same state.

For the definition of *ASK*, we will fix two recursive, non-repetitive enumerations of sentences. First, $P_0, P_1, \ldots$ is an enumeration of some infinite set of tautologies in which variables are the only terms. We will refer to these as the "chosen tautologies." Second, $w_0, w_1, \ldots$ is an enumeration of all the remaining sentences of L, exclusive of $P_0, P_1, \ldots$. Given a sentence w_i among the $w_0, w_1, \ldots$, we will refer to i as the "rank" or "index" of the sentence. For the $P_0, P_1, \ldots$ we could have chosen any infinite recursive set of sentences that still leave infinitely many other sentences, but tautologies are a "tidy" choice, ensuring that a believer does not make inconsistent extrapolations from a consistent database. The beliefs of the belief machine we are constructing will consist of the database, possibly (depending on the state of the machine) augmented by infinitely many of the chosen tautologies, namely $P_j, P_{j+1}, \ldots$ for some $j \geq 0$. The chosen tautologies are defined to have no terms other than variables as a simple way of satisfying the constraint of monotonicity under substitution (C5).

We define the output of *ASK* for database $\{\varphi_1, \ldots, \varphi_n\} (n \geq 0)$ and query ψ, *i.e.* *ASK*$(\{\varphi_1, \ldots, \varphi_n\}, \psi)$, as follows:

- If $\psi \in \{\varphi_1, \ldots, \varphi_n\}$, return yes.
- If $n > 0$ and $\psi = P_j$ for some j, (*i.e.* it is a chosen tautology), then if for some w_i in $\{\varphi_1, \ldots, \varphi_n\}$ (*i.e.* for some sentence in the database that is not a chosen tautology), the ith Turing machine with input 0 halts within j steps, return yes.

[1] This proof is due to Len Schubert.

- If neither of the above conditions applies, return no.

Note that we are using the query ψ, in case it is a chosen tautology, to supply a computation bound j (via its rank) on a a set of Turing machine computations. If those computations never halt, we will return yes only for formulas explicitly in the database, *i.e.* the belief set will be finite. If one of the computations does halt within j steps, then we will return yes for all tautologies from the jth-ranked onward, and so the belief set will be infinite.

We now show that this belief machine satisfies constraints C1–C5. The closure constraint (C1) says that *TELL*ing the machine a sentence it already believes leaves it unaffected. This is clearly true if the sentence is in the database. The only other sentences the machine can believe are the chosen tautologies, from the jth-ranked (for some j) onward. If we *TELL* it such a tautology, no new beliefs are induced since the chosen tautologies occurring within the database are not used for Turing machine computations.

The commutativity and monotonicity constraints (C2 and C3) are obviously satisfied, since all *TELL*ed sentences are accepted (in any order), and the resultant state is independent of order of presentation.

For the acceptable basis constraint (C4), it is clear from the definition of *TELL* and *ASK* that the database of a belief machine provides a set of formulas that is acceptable (in any order) by the machine in the initial state (named by the empty database). Further, *TELL*ing the machine these formulas, starting in the initial state, obviously brings it into the state named by the database.

The constraint of monotonicity under substitution (C5) is satisfied since the belief set of the belief machine contains only sentences that were explicitly *TELL*ed, plus possibly some chosen tautologies, which contain no ground terms to be renamed.

It remains to show that no finite set of recursive inference rules can model this belief machine, in the sense of allowing the same simulative inferences (or "attachment" inferences, as Konolige terms them). This follows from the fact that if there were such a set of inference rules, we could recursively enumerate the Turing machines that do not halt on input 0, which is impossible.

In particular, given recursive inference rules $R_1, \ldots, R_m$, where these derive the same belief set from a given database as *ASK*, we would proceed as follows to enumerate these Turing machines. We dovetail a set of computations corresponding to singleton premise sets $\{w_0\}$, $\{w_1\}, \ldots$, where for each such premise set $\{w_i\}$ we systematically generate all conclusions that can be obtained with rules $R_1, \ldots, R_m$, starting with $\{w_i\}$. In other words, we initiate a deductive closure computation for $\{w_i\}$. For each such closure computation, we continually check whether we have reached closure yet. Note that if a closure computation generates only a finite set of sentences, then we will eventually detect this; viz., at some point we will find that every rule in $R_1, \ldots, R_m$ generates a conclusion (if any) that has already been generated, no matter

how we apply the rule to the conclusions already generated. (There are only finitely many ways of applying an inference rule to a finite set of possible premises, and since the rules are computable functions of their premises, they terminate after some time with a conclusion, or, possibly, with a signal that no conclusion follows for the given premises.)

Whenever we detect closure for the closure computation of some $\{w_i\}$, we add i to the list of indices of the Turing machines that do not halt with input 0. Note that this identification of a non-halting Turing machine is correct, since *ASK* obtains a finite set of beliefs from $\{w_i\}$ only if i is the index of a Turing machine which does not halt for input 0. (If the Turing machine halts, *ASK* will say yes for infinitely many chosen tautologies.) Further, since simulative inference with premise set $\{w_i\}$ using *ASK* will clearly generate *all* beliefs corresponding to database $\{w_i\}$, the rules $R_1, \ldots, R_m$ should also generate all of these beliefs. So clearly the above procedure should effectively enumerate all Turing machines that do not halt with input 0, which is impossible. Hence $R_1, \ldots, R_m$ must not exist. □

However, the requirements of provinciality and effectiveness, as stated by Konolige, are stronger than what is really required, and relaxing them in either of two ways can lead to the equivalence of the two models (with the computational model still subject to the constraints of commutativity, closure, and credulity). One potential weakening is to drop the provinciality constraint, with its implicit assumption that the rule set is finite. Konolige's intent in introducing that restriction appears to have been simply to prohibit default rules, which draw conclusions from the entire knowledge base rather than some subset of a particular size. But the essential feature of default rules that makes them unsuitable is not the variable size of their premise sets, but their defeasibility. As long as the inference rules are monotonic, it would seem that all of Konolige's results still hold, even if infinite rule sets or rules with varying number of premises are allowed.

If we allow deduction structures to contain infinitely many inference rules, then given any belief machine meeting the closure, commutativity, and credulity constraints, one can straightforwardly construct an equivalent set of inference rules: for every possible set of base beliefs and every conclusion the belief machine draws from those beliefs, the set contains an ad hoc rule that licenses that conclusion given those base beliefs. This construction is described more concretely in the proof of the following theorem.

Theorem 17 *If a belief machine satisfies the closure, commutativity, and credulity constraints, then there is an infinite set of inference rules that always yields the same belief set when given the same base beliefs.*

Proof: Given belief machine $m = \langle \Gamma, S_0, \mathit{TELL}, \mathit{ASK} \rangle$, we construct the set of inference rules $R = \{r_1, r_2, \ldots\}$, where r_i is defined as follows: to apply r_i to a set of premises $\varphi_1, \ldots, \varphi_n$,

1. Decode rule index i as a pair of integers $\langle j, k \rangle$, where the encoding is based on a 1-1 recursive function from $\mathbf{N} \times \mathbf{N}$ onto $\mathbf{N}$ (where $\mathbf{N}$ is the set of natural numbers).

2. Decode j as a finite set S of sentences in L, *i.e.* interpret j as the Gödel number of S, based on a Gödel numbering of all finite sets of sentences in L.

3. Decode k as a sentence ψ in L, based on a Gödel numbering of all sentences in L.

4. If $S = \{\varphi_1, \ldots, \varphi_n\}$ and *ASK*(*TELL*$(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$, return conclusion ψ. (Use some fixed method of ordering the φ_i, e.g. lexicographic ordering.)

5. If no conclusion was generated in step 4, signal that the rule is inapplicable to the premises.

Consider an arbitrary base belief set $\{\varphi_1, \ldots, \varphi_n\}$. For conciseness, define

$$B_R = \{\psi | \varphi_1, \ldots, \varphi_n \vdash_R \psi\}$$

and

$$B_m = \{\psi | \textit{ASK}(\textit{TELL}(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes\}$$

In the definition of B_m, the order of the sequence $\varphi_1, \ldots, \varphi_n$ is not significant: m satisfies the commutativity constraint, which says that order is not significant for any acceptable sequence, and the credulity constraint, which says that every sequence is acceptable.

Every sentence $\chi \in B_m$ is also in B_R, because R is constructed to contain the rule $\frac{\varphi_1, \ldots, \varphi_n}{\chi}$. In the other direction, we need to show that for every sentence χ for which there exists a proof $\varphi_1, \ldots, \varphi_n \vdash \chi$, that sentence is in B_m. We will show this by induction on the length of the proof of χ. If the length of the proof is zero, *i.e.* $\chi \in \{\varphi_1, \ldots, \varphi_n\}$, then $\chi \in B_m$ because the credulity constraint says that χ became a belief when it was *TELL*ed to the machine, and that no succeeding *TELL* could cause that belief to be revised. Assume that every sentence provable with the rules R in at most l steps from $\varphi_1, \ldots, \varphi_n$ is in B_m, and that the proof of χ from $\varphi_1, \ldots, \varphi_n$ has $l + 1$ steps. All of the premises used in the last, $l + 1$st step in the proof of χ were themselves proved from $\varphi_1, \ldots, \varphi_n$ in at most l steps, and are therefore in B_m by the induction hypothesis. The rule applied in that $l + 1$st step is the rule $\frac{\psi_1, \ldots, \psi_m}{\chi}$, for some $\psi_1, \ldots, \psi_m$, such that *ASK*(*TELL*$(S_0, \psi_1, \ldots, \psi_m), \chi) = yes$. The credulity constraint says that every sequence is monotonically acceptable, so if *ASK*(*TELL*$(S_0, \psi_1, \ldots, \psi_m), \chi) = yes$ then *ASK*(*TELL*$(S_0, \psi_1, \ldots, \psi_m, \varphi_1, \ldots, \varphi_n), \chi) = yes$. By the commutativity constraint, *ASK*(*TELL*$(S_0, \varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_m), \chi) = yes$. Since each $\psi_i \in \mathcal{B} \cdot$ *TELL*$(S_0, \varphi_1, \ldots, \varphi_n)$, by the closure constraint $\mathcal{B} \cdot$ *TELL*$(S_0, \varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_m) = \mathcal{B} \cdot$ *TELL*$(S_0, \varphi_1, \ldots, \varphi_n)$. Therefore, $\chi \in \mathcal{B} \cdot$ *TELL*$(S_0, \varphi_1, \ldots, \varphi_n)$, *i.e.* $\chi \in B_m$. □

In addition to dropping the provinciality restriction in favor of a prohibition against defeasibility, one could also weaken the effectiveness restriction: instead of requiring the (unique) conclusion to be derivable from the premises by an effectively computable function, we could require only that there be an effectively computable function that decides, given a set of premises and a conclusion, whether the premises justify the conclusion. In that case, given any belief machine satisfying the closure, commutativity, and credulity constraints, there is (trivially) a single inference rule that generates the same belief sets, namely

$$\frac{\varphi_1, \ldots, \varphi_n}{\psi}$$

when *ASK*(*TELL*$(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$.

While this is clearly not the kind of inference rule Konolige had in mind, the weakened restrictions are arguably still reasonable, and none of Konolige's technical results are affected by the change.

To summarize the expressiveness comparison between the two models:

- Agents in the deduction model, but not in the computational model, can have uncomputable belief sets. This is no advantage, since such agents obviously can't exist in practice.

- There are agents in the computational model, but not in the deduction model, for which simulative inference is not sound. In the next section, we argue that this is an advantage for the computational model, since it provides a vocabulary for describing many kinds of inference mechanism, and for expressing constraints that distinguish ones for which simulative inference is sound.

- Agents in the computational model, but not in the deduction model, can choose to reject some of their input sentences, for instance in the case of contradictory inputs. Therefore, for some belief machines a sentence such as $B(a, P(c) \wedge \neg P(c))$ is not satisfiable, while there is no deduction structure for which the analogous sentence is unsatisfiable.

- If we limit consideration to belief machines that satisfy the constraints under which we've shown that that simulative inference is sound, and we add the further restriction that the machine may not reject input sentences even if contradictory, then any remaining belief machine describes an agent that can also exist in the deduction model (but only if we relax Konolige's definition of a deduction structure in a non-trivial way).

5.5.2 Practical Applicability

Our aim in developing a model of belief has been to understand exactly when it is appropriate to add simulative reasoning to a reasoning system, that is, a computer program that manipulates logical formulas. In some cases, Konolige's model can serve that purpose. If the program in question actually works by exhaustively applying a set of deductive inference rules to its inputs, or by performing an exhaustive search for a proof of a query sentence from the input sentences, then the deduction model applies quite straightforwardly. Of course, few reasoning programs work this way. Some systems represent information in forms other than logical formulas, for example using graphical representations that lend themselves to efficient reasoning in particular domains (such as maps for spatial reasoning, graph structures for reasoning about partial temporal ordering, etc.). Even in systems that reason by applying inference rules to logical formulas, the inference rules are typically not applied exhaustively; rather, there is a control mechanism that decides which rule to apply when, and when to give up. For such programs, the deduction model may still apply, but not as straightforwardly. Even if such a program has an equivalent deduction structure, the set of inference rules in that deduction structure is not the same set of rules used by the program itself, since the program doesn't apply those rules exhaustively. If the system can be modeled by a deduction structure at all, then it is one with a relatively complex set of inference rules that incorporate both the system's inference rules and its control structure. The belief machine abstraction is a more straightforward way of describing a program that performs inference.

It's true that to use the deduction model to justify simulative reasoning in a particular program, it is not necessary to list the inference rules of a deduction structure that describes the program's behavior. It is only necessary to show that such a deduction structure exists. But there are programs that are not described by any deduction structure, and Konolige does not discuss how one can distinguish such programs. The belief machine abstraction provides a vocabulary for expressing this question, and the technical results reached in this paper can be used to answer it.

Konolige hints at a technique of augmenting the language of belief to accommodate in the deduction model reasoning methods that violate the closure constraint [p. 24]. His example is to model a reasoner that only discovers conclusions that can be reached from its base beliefs in at most n applications of *modus ponens*, for some fixed n. The technique is to add a "depth predicate" D to the language, and give the agent the following deduction rule:

$$\frac{D(k) \wedge \varphi, \qquad D(l) \wedge (\varphi \supset \psi)}{D(k+l+1) \wedge \psi} \text{ for } k + l + 1 \leq n.$$

This proposal is simply not consistent with the semantics of Konolige's logic. Deduction structures are supposed to use only sound inference rules, but the above rule is clearly not sound, unless we take the agent being modeled to be using a non-standard

semantics, one that restricts the interpretation of the predicate D, which is different from the semantics by which our own beliefs are interpreted. This is, in fact, left implicitly as a possibility [p. 32], but it is a distasteful one—if the sentences we attribute as the beliefs of others are interpreted with a different semantics than the sentences of our own beliefs, then how are we to understand what beliefs we are attributing? Furthermore, once the predicate D has been introduced into the language of belief, sentences such as $B(a, D(5) \wedge P(b) \wedge D(8))$ and $B(a, D(5))$ are syntactically well-formed, even though they have no meaning in the proposed scheme.

5.5.3 Completeness in the Deduction Model

In Section 3.4.1, we showed that our logic is incomplete, *i.e.* given certain belief machines, there are formulas that are unsatisfiable yet cannot be disproved. In contrast, Konolige proves a general completeness theorem, which holds given an arbitrary choice of inference rules, for his deduction model. Since the concepts of the belief machine and the deduction structure are quite similar, it is natural to ask why there are belief machines for which our logic is incomplete, but there are no corresponding sets of inference rules for which Konolige's is incomplete. The answer, as we will now show, is that completeness in Konolige's model depends on the fact that the base belief set may be infinite (the attachment lemma [Lemma 3.3] depends on this), while our model requires that there be only finitely many base beliefs. At a given time, a real agent can only have explicitly learned finitely many facts, so it is unrealistic to allow the base belief set to be infinite.

We noted in Section 3.4.1 that our proof theory is incomplete given a belief machine that answers yes only to those sentences it has already been *TELL*ed, because given such a machine, the sentence $\forall x B(a, P(x))$ has only finite models. In Konolige's model, there is a set of inference rules, namely the empty set, that makes the same deductions as our example belief machine; but given that set of inference rules, the sentence $\forall x B(a, P(x))$ has both finite and infinite Konolige-style models. In the infinite models, the agent denoted by a simply has infinitely many base beliefs.

This shows that the difference in completeness between our logic and Konolige's is not very significant. The difference is simply that in Konolige's logic, certain theories that intuitively should be unsatisfiable are satisfiable and consistent, whereas in our logic those theories are unsatisfiable but still consistent.

5.5.4 Other Differences

In addition to modeling inference in a different way, we have made some other choices in the design of our representation that differ from Konolige's. These other differences are orthogonal to the more fundamental choice of how to model inference.

ID constants: Konolige requires that for each agent, there be a naming map that maps each individual in the domain to a unique *id constant*. In other words, while an agent may use several different terms to refer to the same individual, one of those terms is always distinguished as the canonical name. ID constants are used in the semantics of quantifying-in: an open belief formula $\exists x B(a, P(x))$ (translating Konolige's notation into ours) is true iff there is some id constant κ for which B(a,P(κ)) is true. We have chosen not to require that individuals have canonical names, and we have defined the semantics so that a quantified-in formula $\exists x B(a, P(x))$ is true if there is any term τ for which $B(a, P(\tau))$ is true.

Quantification over believers: In Konolige's logic, there is a different belief operator for each agent, rather than a single operator that takes a believer argument. Therefore, quantification over believers is impossible in Konolige's logic—there is no equivalent of the formula $\forall x B(x, P(c))$ of our logic.

Equality: Our logic includes an equality operator, while Konolige's does not. This is one of the things that makes the (restricted) completeness proof for our logic more complex.

6 Conclusions

The concept of belief is an essential part of our intuitive understanding of human behavior. If we are to design a machine whose ability to reason about and interact with humans approaches our own, we should make it able to reason about humans in terms of their beliefs. This means we must provide it with a predictive model of human belief. This is certainly not an easy task, but in a sense it is the very aim of all AI research. The idea of simulative inference is that in order to reason about humans, an AI system can use itself as a model of human belief.

This is not a new idea. Forms of simulative inference have been described in the AI literature at least since [Moore, 1973]. But, while the intuition behind simulative inference is clear, we feel that prior to our work, the intuition had not been transformed into an adequate formal semantics for belief.

The role of semantics in an AI system is to define the meaning of the representations the system uses, thereby serving as justification for inferences the system makes. Historically, simulative inference has often been applied without any formal basis, but there have been a few previous attempts to define belief precisely and in a way that justifies simulative inference. One approach was to use Hintikka's possible worlds model, but this is well-known to be problematic. It requires that agents' beliefs be closed under logical consequence, even in cases where the closure isn't computable, and it denies the possibility of an agent that maintains contradictory beliefs in a useful way. Levesque and others have modified the possible worlds model to weaken logical omniscience while retaining a characterization of belief sets as closed under some form of consequence. This approach has arguably been successful at defining a "lower bound" for belief, by stipulating, for example, that believing $\varphi \wedge \psi$ is the same as believing $\psi \wedge \varphi$. But in these models, many reasonable and desirable entailments are eliminated along with the unreasonable ones. They don't predict consequences which intuitively we know exist, for example that someone who believes that Fido is a doberman and believes that dobermans are dogs also believes that Fido is a dog.

Konolige responded to these theories with the claim that belief has the characteristics it does because beliefs are held by a believer, which (the field of AI holds) is a computational entity. The properties of belief depend crucially on the properties of the

computation that entity performs. Therefore, Konolige laid out a semantics of belief in which a believer is modeled as a particular sort of computational mechanism, namely a set of inference rules which are applied exhaustively.

Konolige's model is important for acknowledging that computation is a crucial part of the definition of belief. However, the class of models of believers allowed by Konolige's theory is quite limited. AI programs are intended to be models of believers, but many such programs can't be described in the terms the deduction model provides. Furthermore, it isn't always clear whether a given program can be described in those terms or not.

Therefore, we have set out to define a theory of belief that allows a much more general class of believer models. Konolige's approach was to use a very restrictive definition of a model of a believer, and to prove that simulative inference was sound for all such models. Our approach was to use a much more general definition, so general that simulative inference is clearly *not* always sound, and then to ask what properties the believer's computation must have in order for simulative inference to be sound.

Perhaps not surprisingly, the conditions under which simulative inference is sound turn out to be ones that many AI programs don't satisfy. It will rarely be the case that one can examine a system and simply declare, thanks to our results, that simulative inference performed by the system is correct. The value of our work, therefore, is in that it clearly and explicitly states the assumptions underlying simulative inference, assumptions which before now had often remained implicit.

The case study of simulative inference in EPILOG demonstrated how our model can be of use in the principled design of a reasoning system. We found that EPILOG violated the conditions for sound simulative inference in many ways. When a violation is discovered, one can modify the system so that it no longer performs the problematic sort of computation, at least not during simulations when it is being used as a model of a believer, or one can simply show that problematic cases will not arise in practice.

Our work on EPILOG also brought to light a problem we had not foreseen for the efficient implementation of simulative inference. In fact, the problem is not specifically related to belief or simulation, but to the more general and widely-studied technique of reason maintenance. What we discovered is that in a reasoning system that, like EPILOG, stores information in various non-sentential forms, reason maintenance is difficult to implement without destroying the efficiency advantage the special representations are designed to provide. This problem has undoubtedly confronted many system designers before in various guises, but we are not aware of its previously having been identified in such general terms.

6.1 Future Work

We have seen that real reasoning systems often violate the conditions for correct simulative inference. Some of these violations suggest directions for refinement of the model, to handle in a more principled way the situations in which the conditions are not met. For example, we acknowledged the fact that, in practice, different people have different inferential ability, but we appealed to the intuition that there is a basic level of ability that is common to all. We also acknowledged that current AI systems are far from perfect models of human belief, but appealed to the intuition that they usually derive a subset of the beliefs a human would reach from the same information, so the difference between the model and reality is not likely to cause simulative inference to give incorrect results. Both of these intuitions could be explored more formally by allowing different agents in the model to have different belief machines, and defining what it means for one machine to subsume another.

The monotonicity constraint is one that AI systems often don't satisfy, and this too suggests directions for future work. We have shown how with negative introspective machines, simulative inference can essentially take into account information about what an agent is known *not* to believe, as well as what it is known to believe, thereby allowing monotonic simulative inference using a nonmonotonic belief machine. Levesque's [1990] logic of "only knowing" does this too, but is more flexible: rather than specifying particular sentences that an agent doesn't believe, the logic allows one to circumscribe what the agent does believe. It has an operator whose meaning is "All α knows is φ," *i.e.* for any sentence ψ, α doesn't know ψ unless knowing ψ is a consequence of knowing φ. This operator could be adapted to our computational model, and a form of simulative inference that uses "only knowing" information could be defined. This kind of information might seem difficult to obtain (how can we know *everything* another person knows?), but [Lakemeyer, 1995] refines the idea of "All α knows is φ" to "All α knows that is relevant to ψ is φ;" our computational framework could be used to give a very satisfying account of relevance.

The notion of *common knowledge*, information that all members of a group know, and know that they all know, *etc. ad infinitum*, arises in various areas of AI, for example in theories of the *common ground* of a dialog. The possible worlds model is often used in the semantics of common knowledge, and is problematic there for the same reasons as for ordinary belief. This could be fertile ground for a computational account.

Reasoning about belief is not an end in itself. It is useful because it allows us to explain and predict behavior. We stated glibly at the beginning of this chapter that building a model of human belief is the aim of all AI research, but it would be more accurate to say that the aim of AI research is to build a unified model of human perception and action. Internally, such a model will undoubtedly incorporate not only the concept of belief, but also other mental states such as desires and intentions. An AI system that models many of these states could reason about all of them at once by simulation.

It would be interesting to extend our theory of belief to a more general computational theory of agency.

The term *rational* is used in economics and sociology, particularly in game-theoretic approaches, to describe an agent who always acts in a way that will maximize the expected value for him of the resulting situation. The idealization of rationality is closely related to that of logical omniscience, and is unrealistic for the same reasons. A computational model of belief, and perhaps of other propositional attitudes, might be useful in theories of "bounded rationality."

Our implementation of simulative inference in EPILOG is at a rather preliminary stage, and there are many things that could be done to improve it. Perhaps the most important missing functionality is the ability to reason about belief change. Another issue of primary concern is the design of the interestingness criterion. Input-driven inference predicts what a human will believe after learning a given series of facts, regardless of whether he has other reasons to ask himself particular questions. It therefore provides a way to build an inference mechanism that is non-trivial but satisfies the closure constraint. The interestingness metric should be designed to maintain commutativity of consistent inputs, in order to be compatible with simulative inference.

Bibliography

[Allen and Miller, 1991] James F. Allen and Bradford W. Miller, "The RHET System: A Sequence of Self-Guided Tutorials," Technical Report 325, University of Rochester, Rochester, NY, July 1991.

[Ballim and Wilks, 1991] Afzal Ballim and Yorick Wilks, *Artificial Believers (The Ascription of Belief)*, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1991.

[Barwise and Perry, 1983] J. Barwise and J. Perry, *Situations and Attitudes*, MIT Press, Cambridge, MA, 1983.

[Braun, 1998] David Braun, "Understanding Belief Reports," *Philosophical Review*, 1998.

[Buvač, 1993] Saša Buvač, "Propositional Logic of Context," In Richard Fikes and Wendy Lehnert, editors, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 412–419, Menlo Park, California, 1993. American Association for Artificial Intelligence, AAAI Press.

[Carnap, 1947] R. Carnap, *Meaning and Necessity*, University of Chicago Press, Chicago, 1947.

[Chalupsky, 1996] Hans Chalupsky, *SIMBA: Belief Ascription by Way of Simulative Reasoning*, PhD thesis, Department of Computer Science, State University of New York at Buffalo, 1996.

[Chalupsky and Shapiro, 1996] Hans Chalupsky and Stuart C. Shapiro, "Reasoning About Incomplete Agents," In *Proceedings of the Fifth International Conference on User Modeling*, 1996.

[Church, 1950] Alonzo Church, "On Carnap's Analysis of Statements of Assertion and Belief," *Analysis*, 10(5):97–99, 1950.

[Creary, 1979] Lewis G. Creary, "Propositional Attitudes: Fregean Representation and Simulative Reasoning," In *IJCAI-79: Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, volume 1, 1979.

[Cresswell, 1985] M. J. Cresswell, *Structured Meanings: The Semantics of Propositional Attitudes*, MIT Press, Cambridge, Massachusetts, 1985.

[Delgrande, 1995] James P. Delgrande, "A Framework for Logics of Explicit Belief," *Computational Intelligence*, 11(1):47–88, 1995.

[Dinsmore, 1991] John Dinsmore, *Partitioned Representations*, Kluwer Academic Publishers, 1991.

[Eberle, 1974] Rolf A. Eberle, "A Logic of Believing, Knowing, and Inferring," *Synthese*, 26:356–382, 1974.

[Elgot-Drapkin and Perlis, 1990] Jennifer J. Elgot-Drapkin and Donald Perlis, "Reasoning situated in time I: basic concepts," *Journal of Experimental and Theoretical Artificial Intelligence*, 2:75–98, 1990.

[Fagin and Halpern, 1988] Ronald Fagin and Joseph Y. Halpern, "Belief, Awareness, and Limited Reasoning," *Artificial Intelligence*, 34:39–76, 1988.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, *Reasoning about Knowledge*, MIT Press, Cambridge, Massachusetts, 1995.

[Fodor, 1975] Jerry A. Fodor, *The Language of Thought*, Thomas Y. Crowell Company, New York, 1975.

[Frege, 1892] Gottlob Frege, "Über Sinn und Bedeutung," *Zeitschrift für Philosophie und Philosophische Kritik*, 100:25–50, 1892, Translated as "On sense and reference" in [Frege, 1977].

[Frege, 1977] Gottlob Frege, *Translations from the Philosohpical Writings of Gottlob Frege*, Blackwell, Oxford, 1977, trans. P. T. Geach and M. Black.

[Gerevini *et al.*, 1995] A. Gerevini, L. K. Schubert, and S. Schaeffer, "The temporal reasoning tools TimeGraph I-II," *International Journal of Artificial Intelligence Tools*, 4(1 and 2):281–299, 1995.

[Giunchiglia *et al.*, 1993] Fausto Giunchiglia, Luciano Serafini, Enrico Giunchiglia, and Marcello Frixione, "Non-omniscient belief as context-based reasoning," In *Thirteenth International Joint Conference on Artificial Intelligence*, 1993.

[Grove, 1995] Adam J. Grove, "Naming and Identity in Epistemic Logic Part II: a First-Order Logic for Naming," *Artificial Intelligence*, pages 311–350, 1995.

[Haas, 1986] Andrew R. Haas, "A Syntactic Theory of Belief and Action," *Artificial Intelligence*, 28:245–292, 1986.

[Haas, 1995] Andrew R. Haas, "An Epistemic Logic with Quantification over Names," *Computational Intelligence*, 1995.

[Halpern *et al.*, 1994] Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, "Algorithmic Knowledge," In Ronald Fagin, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pages 255–266, San Francisco, 1994. Morgan Kaufmann.

[Hintikka, 1962] Jaakko Hintikka, *Knowledge and Belief*, Cornell University Press, Ithaca, New York, 1962.

[Hodges, 1983] Wilfrid Hodges, "Elementary Predicate Logic," In D. Gabbay and F. Guenthner, editors, *Elements of Classical Logic*, volume I of *Handbook of Philosophical Logic*, chapter 1. D. Reidel, Boston, 1983.

[Hughes and Cresswell, 1968] G. E. Hughes and M. J. Cresswell, *An Introduction to Modal Logic*, Methuen, London, 1968.

[Hwang and Schubert, 1993] Chung Hee Hwang and Lenhart K. Schubert, "Episodic Logic: A Comprehensive, Natural Representation for Language Understanding," *Minds and Machines, Special Issue on KR in NLP*, 3(4):381–419, 1993.

[Isozaki, 1995] Hideki Isozaki, "Reasoning About Belief Based on Common Knowledge of Observability of Actions," In Victor Lesser, editor, *First International Conference on Multi-Agent Systems*, pages 193–200. AAAI Press / The MIT Press, 1995.

[Jackendoff, 1983] Ray Jackendoff, *Semantics and Cognition*, MIT Press, Cambridge, Massachusetts, 1983.

[Jameson, 1995] Anthony Jameson, "Logic is Not Enough: Why Reasoning About Another Person's Beliefs Is Reasoning Under Uncertainty," In Armin Laux and Heinrich Wansing, editors, *Knowledge and Belief in Philosophy and Artificial Intelligence*. Akademie Verlag, Berlin, 1995.

[Kaplan, 1969] David Kaplan, "Quantifying in," In D. Davidson and K. J. J. Hintikka, editors, *Words and Objections: Essays on the Work of W. V. Quine*, pages 206–242. Reidel, Dordrecht, 1969.

[Konolige, 1986] Kurt Konolige, *A Deduction Model of Belief*, Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

[Kripke, 1963] S. A. Kripke, "Semantical Considerations on Modal Logic," *Acta Philosophica Fennica*, 16:83–94, 1963.

[Kripke, 1980] Saul Kripke, *Naming and Necessity*, Blackwell, Oxford, 1980.

[Lakemeyer, 1994] Gerhard Lakemeyer, "Limited reasoning in first-order knowledge bases," *Artificial Intelligence*, 71:213–255, 1994.

[Lakemeyer, 1995] Gerhard Lakemeyer, "Relevance in a Logic of Only Knowing About and its Axiomatization," In Armin Laux and Heinrich Wansing, editors, *Knowledge and Belief in Philosophy and Artificial Intelligence*. Akademie Verlag, Berlin, 1995.

[Lakemeyer, 1997] Gerhard Lakemeyer, "Relevance From an Epistemic Perspective," *Artificial Intelligence*, 97(1–2):137–167, 1997.

[Lenat and Guha, 1990] D. B. Lenat and R. V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*, Addison-Wesley, Reading, Massachusetts, 1990.

[Levesque, 1984] Hector. J. Levesque, "A Logic of Implicit and Explicit Belief," In *Proceedings AAAI-84*, pages 198–202, Austin, 1984.

[Levesque, 1990] Hector J. Levesque, "All I Know: A Study in Autoepistemic Logic," *Artificial Intelligence*, 42:263–309, 1990.

[McCarthy, 1958] John McCarthy, "Programs With Common Sense," In *Proceedings of the Teddington Conference on the Mechanisation of Thought Processes*, 1958, Reprinted in [Minsky, 1960].

[McCarthy, 1993] John McCarthy, "Notes on Formalizing Context," In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 81–98, Los Altos, California, 1993. Morgan Kaufmann.

[McDermott and Sussman, 1972] Drew McDermott and G. J. Sussman, "The Conniver Reference Manual," Technical Report 259, MIT, 1972.

[Miller and Schubert, 1990] S. A. Miller and L. K. Schubert, "Time Revisited," *Computational Intelligence*, 6(2):108–118, 1990.

[Minsky, 1960] Marvin Minsky, editor, *Semantic Information Processing*, MIT Press, Cambridge, MA, 1960.

[Montague, 1973] Richard C. Montague, "The Proper Treatment of Quantification in Ordinary English," In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*. Reidel, Dordrecht, 1973.

[Moore, 1973] Robert C. Moore, "D-SCRIPT: A Computational Theory of Descriptions," In *Third International Joint Conference on Artificial Intelligence*, pages 223–229, 1973.

[Moore, 1977] Robert C. Moore, "Reasoning About Knowledge and Action," In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 1, pages 223–227, 1977.

[Moore, 1995] Robert C. Moore, *Logic and Representation*, CSLI Lecture Notes. Center for the Study of Language and Information, 1995.

[Moore and Hendrix, 1982] Robert C. Moore and Gary G. Hendrix, "Computational Models of Belief and the Semantics of Belief Sentences," In S. Peters and E. Saarinen, editors, *Processes, Beliefs, and Questions*. D. Reidel, Dordrecht, Netherlands, 1982, reprinted in [Moore, 1995].

[Perlis, 1988] Donald Perlis, "Languages with Self-Reference II: Knowledge, Belief, and Modality," *Artificial Intelligence*, 34:179–212, 1988.

[Rapaport *et al.*, 1997] William J. Rapaport, Stuart C. Shapiro, and Janyce M. Wiebe, "Quasi-Indexicals and Knowledge Reports," *Cognitive Science*, 21(1):63–107, 1997.

[Salmon, 1986] Nathan Salmon, *Frege's Puzzle*, MIT Press, Cambidge, MA, 1986.

[Schubert and Hwang, 2000] Lenhart K. Schubert and Chung Hee Hwang, "Episodic Logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding," In L. Iwanska and S.C. Shapiro, editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. MIT/AAAI Press, 2000.

[Soames, 1988] Scott Soames, "Direct Reference, Propositional Attitudes, and Semantic Content," In Nathan Salmon and Scott Soames, editors, *Propositions and Attitudes*. Oxford University Press, Oxford, 1988.

[Stalnaker, 1984] Robert Stalnaker, *Inquiry*, MIT Press, Cambridge, Massachusettes, 1984.