ELSEVIER

Artificial Intelligence 120 (2000) 119–160

Artificial Intelligence

www.elsevier.com/locate/artint

A computational model of belief

Aaron N. Kaplan *, Lenhart K. Schubert [1]

Computer Science Department, University of Rochester, Rochester, NY 14627-0226, USA

Received 9 June 1999; received in revised form 9 March 2000

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. We also give a detailed comparison of our model with Konolige's deduction model, another model of belief in which the believer's reasoning mechanism is a parameter. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Belief; Belief inference; Belief ascription; Omniscience; ASK/TELL mechanism; Simulative reasoning; Computational model of belief; Deduction model of belief

1. Introduction

If an observer sees dark clouds and hears thunder, it might cause him to believe that it will rain soon. Similarly, if the observer knows that Mary sees clouds and hears thunder, he might conclude that she believes it will rain. The observer draws this conclusion by drawing an analogy between Mary's mental processes and his own: by observing how his own beliefs work, he sees that if he were in her position, he would believe rain was coming.

This kind of reasoning, called *simulative reasoning*, has been explored in the AI literature (e.g., [2,3]), but with the exception of [17], which we discuss below, the treatments have lacked precise descriptions of conditions under which simulative reasoning yields all and only correct conclusions.

* Corresponding author. Email: kaplan@cs.rochester.edu.

[1] Email: schubert@cs.rochester.edu.

0004-3702/00/$ – see front matter
PII: S0004-3702(00)00021-7

In order to make precise arguments of this kind, we need a formal model of belief. The classical "possible worlds" model [18] might seem a natural choice, because it has the following property: in the possible worlds model, if an agent believes some proposition φ, and φ logically entails ψ, then the agent also believes ψ. If the reasoning performed in a simulation is sound, then any conclusion reached by simulative reasoning is justified by the possible worlds model. But there are two problems with this model. First, believers can only perform sound inference, while real believers may use unsound reasoning techniques such as default reasoning. We will not address this issue in the current paper, but it is the subject of [13]. Second, in the possible worlds model, believers are "logically omniscient". This means that if some proposition follows from an agent's beliefs, then, according to the model, the agent believes that proposition, no matter how difficult it might be to discover. This violates common intuitions about what it means to believe something. We take "α believes φ" to mean that agent α can see with no conscious effort that φ is true given what he has learned. If something does follow from what α has learned, but it would take a significant amount of reflection to discover this connection, then we do not say that α believes φ, only that after sufficient reflection he could come to believe it. This view seems to us consistent with ordinary usage of the word "believes" (and similarly "knows"). For example, most people who can correctly answer simple questions such as "Is Copenhagen north of Rome?", "Is twelve a prime number?", or "Are squirrels primates?" will also maintain that they have known (and thus believed) the correct answers for a long time, even though they may never have explicitly considered those questions. By contrast, most people who can correctly answer "Is 97 a prime number?" or "Are the full names of Abraham Lincoln's and John Kennedy's assassins equal in length?" will do so only after some deliberate reflection, and the longer they take to answer, the less likely they are to claim that they knew the answer all along. Rather, they will say that they worked it out, and now know it.

The idealized form of belief described by the possible worlds model has come to be known as "implicit belief" (terminology introduced by Levesque [20]), since it describes what is implicit in what an agent believes. This is in contrast to "explicit belief", which describes what an agent actively holds to be true. Much of the past work on explicit belief has involved refinements to the possible worlds model that weaken the logic of belief in various ways, eliminating some entailments that are inappropriate for explicit belief. But such models are still far from capturing the ordinary, non-technical meaning of the word "believes". If a person believes that Fido is a terrier, and that terriers are dogs, then clearly he also believes that Fido is a dog. In eliminating logical omniscience, many logics of explicit belief also lose the ability to predict this inference.

The notion of belief we wish to capture involves the question of what inferences a believer makes automatically, with no conscious effort. It is not surprising that this notion isn't captured by logics in which belief is characterized via semantic constraints on the sentences or propositions that are believed. The inferences a believer makes easily and automatically are determined by the mechanism he uses for maintaining his beliefs. This mechanism may be very complex indeed, and there is no reason to think that its effects could be described by a simple set of semantic constraints.

In this paper, we describe a model of belief that includes an explicitly computational model of information storage and retrieval. In the model, each believer has a *belief*

machine, which is a computational mechanism capable of checking current beliefs and accepting new ones. 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 $ASK(S, \varphi)$ 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 $TELL(S, \varphi)$. It may or may not believe φ in this new state, i.e., it may or may not accept the proffered fact.

Obviously, we do not intend to give in this paper 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 storage and retrieval, there is a logic for reasoning about the beliefs of agents who use that mechanism. Unlike many previous authors, who have imposed a particular inference mechanism and then explored the consequences of that choice, we prove that our logic has certain desirable properties no matter what mechanism is used, as long as it satisfies a particular set of formal (and intuitively natural and rather easily verifiable) constraints.

In order to study simulative reasoning, 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 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.

The paper is organized as follows. In Section 2, we define a logic whose semantics incorporates the computational model of belief. In Section 3.1, we describe simulative reasoning as an inference rule in this logic, and show that the rule is sound under certain constraints on the functions *ASK* and *TELL*. In Section 3.3, we present some more inference rules, and show conditions on the belief machine under which they are sound. Then, in Section 4, we treat the issue of completeness. We show that for some choices of belief machine, no complete set of inference rules can exist; however, we also show that the set of rules presented in this paper is refutation complete for a syntactically restricted subset of the logic (roughly stated, sentences that don't use universal quantifying-in), given any belief machine for which the rules are sound.

The axioms of the classical modal logics **T**, **S4**, **S5**, etc., and other axioms such as the Barcan formula, describe various properties of belief that hold in certain models. These axioms serve as standard criteria for evaluating and comparing modal logics. In Section 5, we discuss several of the classical axioms, and characterize the kind of computation the belief machine must perform in order for each them to be valid.

In Section 6, we compare our computational model of belief with other models. We mention various semantic characterizations of belief, but the greater part of the section is devoted to Konolige's deduction model [17], which is like our model in that it acknowledges that inference is a crucial component of belief, and in that it licenses a form of simulative reasoning. We point out the differences between our model and that

of Konolige, and examine how these differences affect the relative expressiveness and appropriateness of the two models.

2. Syntax and semantics

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 then poses queries to it. Input and queries to the machine 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 $TELL(S, \varphi)$ is the new state the belief machine will enter after φ is asserted to it. The value of $ASK(S, \varphi)$ 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 $\langle \Gamma, S_0, TELL, ASK \rangle$, where

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

Formulas of L have 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 $m = \langle \Gamma, S_0, TELL, ASK \rangle$, 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,
- $\gamma : D \to \Gamma$ 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 $\forall v\varphi$ 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 v; and similarly for existential formulas.

A *variable substitution* is a mapping from variables to ground terms. A variable substitution σ is *extension-preserving* under model M if, for every variable v, the denotation of v and the denotation of $\sigma(v)$ 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 $m = \langle \Gamma, S_0, \mathit{TELL}, \mathit{ASK}\rangle$ 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 $\mathit{ASK}(\gamma(|\alpha|^M), \varphi^\sigma) = \mathit{yes}$. This gives a variable that occurs in a belief context, but is not bound in that context, a reading of implicit existential quantification over term names. 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.

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

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

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.,

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

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

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

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

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

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 $\mathit{TELL}(S, \varphi_1, \ldots, \varphi_{i-1})$.

3. Inference rules

Some inference rules from ordinary FOL are also sound in our logic. Others are sound for some choices of belief machine, but not for others. In this section, we present a set of natural deduction-style inference rules for the logic, and show that given certain constraints on the belief machine, they are all sound.

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

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 as Theorem 1 that it is sound when the belief machine satisfies the constraints listed below. Though the inference rule only explicitly requires that

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

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.

The simulative inference rule is sound under the following three constraints:

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

$$ASK(S, \varphi) = yes$$

then

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

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

C2 (**finite basis**). *For any belief state S, there exists a monotonically acceptable sequence of sentences $\varphi_1, \ldots, \varphi_n$ such that*

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

The finite 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, monotonically

acceptable sequence of sentences. It requires that even if the belief machine reached its current state via a sequence of *TELL*s that was not monotonically acceptable, the same beliefs can be induced via some monotonically acceptable sequence.

Keep in mind that the belief machine need not describe an agent's entire reasoning capability. It is intended to describe only the kinds of inference the agent makes easily and automatically. If *TELL*ing φ to an agent's belief machine would cause its belief set to change, the closure constraint does not preclude the possibility that the agent might discover that φ follows from its beliefs. It only requires that the *belief machine* can't make this discovery, i.e., that the agent doesn't currently believe φ. The agent may have reasoning facilities external to the belief machine that it can use to discover φ. If it did so, it would presumably update its belief machine with the new information, consequently entering a new belief state. In that case, we would say that the agent's beliefs changed as a result of reflection.

C3 (**commutativity**). *For any belief state S and monotonically 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 monotonically acceptable, and*

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

The commutativity constraint says that if a sequence of sentences is monotonically acceptable, then it is monotonically acceptable in any order, and the belief set of the resulting state does not depend on the order. Note that this constraint does permit the belief machine to take order into account when deciding how to handle contradictory (i.e., non-monotonically-acceptable) inputs.

Theorem 1 states that the simulative inference rule is sound given any belief machine that satisfies the above three constraints.

Theorem 1 (Soundness of simulative inference). *For belief machine*

$$m = \langle \Gamma, S_0, TELL, ASK \rangle$$

satisfying constraints (C1)–(C3), *m-model $M = \langle D, I, \gamma \rangle$, and sentences $\varphi_1, \ldots, \varphi_n$ and ψ, if $M \models \bigcup_i \{B(\alpha, \varphi_i)\}$, and $ASK(TELL(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes$, then $M \models B(\alpha, \psi)$.*

Proof. Assume that

$$M \models B(\alpha, \varphi_i) \tag{1}$$

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

$$ASK(TELL(S_0, \varphi_1, \ldots, \varphi_n), \psi) = yes. \tag{2}$$

Let $S = \gamma(|\alpha|^M)$, i.e., the belief state of the agent denoted by α. From (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

$$ASK(S, \varphi_i^{\sigma_i}) = yes. \tag{3}$$

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) is equivalent to

$$ASK(S, \varphi_i) = yes, \quad 1 \leqslant i \leqslant n. \tag{4}$$

According to the finite basis constraint (C2), there is some sequence of sentences $\chi_1, \ldots, \chi_m$ which is monotonically acceptable for S_0, such that

$$\mathcal{B} \cdot TELL(S_0, \chi_1, \ldots, \chi_m) = \mathcal{B} \cdot S. \tag{5}$$

From (4) and (5), it follows that

$$ASK(TELL(S_0, \chi_1, \ldots, \chi_m), \varphi_i) = yes, \quad 1 \leqslant i \leqslant n. \tag{6}$$

From (6) and n successive applications of the closure constraint (C1), it follows that

$$\mathcal{B} \cdot TELL(S_0, \chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n) = \mathcal{B} \cdot TELL(S_0, \chi_1, \ldots, \chi_m), \tag{7}$$

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

$$\begin{aligned}
&\mathcal{B} \cdot TELL(S_0, \varphi_1, \ldots, \varphi_n, \chi_1, \ldots, \chi_m) \\
&\quad = \mathcal{B} \cdot TELL(S_0, \chi_1, \ldots, \chi_m, \varphi_1, \ldots, \varphi_n) \\
&\quad = \mathcal{B} \cdot TELL(S_0, \chi_1, \ldots, \chi_m) \quad \text{(see (7))} \\
&\quad = \mathcal{B} \cdot S \quad \text{(see (5))}.
\end{aligned} \tag{8}$$

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

$$\mathcal{B} \cdot TELL(S_0, \varphi_1, \ldots, \varphi_n) \subseteq \mathcal{B} \cdot TELL(S_0, \varphi_1, \ldots, \varphi_n, \chi_1, \ldots, \chi_m). \tag{9}$$

Eq. (2) says that

$$\psi \in \mathcal{B} \cdot TELL(S_0, \varphi_1, \ldots, \varphi_n) \tag{10}$$

From (8), (9), and (10), it follows that

$$\psi \in \mathcal{B} \cdot S \tag{11}$$

or, equivalently,

$$ASK(S, \psi) = yes. \tag{12}$$

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:

$$M \models B(\alpha, \psi). \qquad \square \tag{13}$$

3.2. An example belief machine

The constraints under which we have proved the soundness of simulative inference are not so restrictive as to exclude interesting kinds of computation. We will now illustrate this with one example. 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

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

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 finite 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.

3.3. Other rules

We now give the remainder of the set of inference rules that will be proved complete in Section 4.

3.3.1. 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, $1 \leqslant i \leqslant n$.

This rule says that if *TELL*ing the belief machine a set of sentences doesn't cause it to believe all of those sentences, then no agent can believe all of them simultaneously. It is sound for any belief machine that satisfies constraints (C1)–(C3).

Theorem 2 (Soundness of negative simulative inference). *Given a belief machine that satisfies constraints* (C1)–(C3)*, if*

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

for some $1 \leqslant i \leqslant n$*, 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 $ASK(S,\varphi_i) = yes$ for all $1 \leqslant i \leqslant n$. By the finite basis constraint, there is some sequence of sentences $\psi_1,\ldots,\psi_m$ that is monotonically acceptable for S_0, such that

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

This means that

$$ASK(TELL(S_0, \psi_1, \ldots, \psi_m), \varphi_i) = yes, \quad 1 \leqslant i \leqslant n.$$

By the closure constraint, if we take a machine in state $TELL(S_0, \psi_1, \ldots, \psi_m)$ 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 monotonically acceptable for S_0. Therefore, the commutativity constraint applies; it says that the sequence

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

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

3.3.2. Rules of propositional logic

The remainder of the inference rules apply to all formulas, not just closed ones.

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 $\supset$.[2]

- 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

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

- Reductio ad absurdum

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

[2] 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.

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.

3.3.3. 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:

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

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:

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

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 3 (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. □

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

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

3.3.4. 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.

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

Lemma 4. *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,

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

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 5 (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 4 it follows that $M \models \varphi_{\tau/\nu}$. □

3.3.5. $\exists$-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 $\exists$-elimination in those terms:

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

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 $\exists$-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 $\exists$-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 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 $\exists$-elimination uses free variables instead of ground Skolem terms, it avoids drawing unjustified conclusions. However, if this were the only form of $\exists$-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 $\exists$-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:

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

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 constraint formalizes this requirement; we will then prove that for machines that satisfy the finite basis constraint (C2) and the new constraint, the belief Skolemization rule is sound.

C4 (**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*

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

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

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

(2) *if*

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

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

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

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

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 6 (Soundness of proofs). *Given a belief machine m satisfying constraints* (C1)–(C4)*, 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 $\leqslant n$ is sound. A proof of ψ from Γ of length $n+1$ is composed of a proof of length $\leqslant n$, 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 than $\exists$-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 $\leqslant n$ 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 $\leqslant n$. $M \models \Delta \cup \{B(\alpha, \varphi)\}$ 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 (C4) 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 Δ, α, φ, 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 $\leqslant n$, 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. □

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 [17] presents a logic of belief that is similar to ours, but for which a general completeness proof does exist. Section 6.1.3 points out an unrealistic assumption that weakens the notion of entailment in Konolige's model, making the completeness proof possible.

4.1. General incompleteness

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 7. *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 8. *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 7. 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. □

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 3.3.4, to make the definition of a positive embedding precise, we treat $\exists v\psi$ as an abbreviation for $\neg\forall v\neg\psi$, 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 9 (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 [9]. 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 $i \geqslant 0$, 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 $\neg\forall v\chi$, and there is no variable μ for which $\neg\chi_{\mu/v}$ is already in Δ_i, then $\neg\chi_{\mu/v}$ 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 $\forall v \chi$, then $\chi_{\tau_j/v}$ is in Δ_{i+1} for the first term τ_j from the ordering chosen above for which $\chi_{\tau_j/v}$ 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 $\neg\chi_{\mu/v}$ introduced in case (6) is not licensed as a sound inference, but if $\Delta_i \cup \{\neg\chi_{\mu/v}\} \vdash \bot$, 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 v 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 $\bot$ 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 $i \geqslant 1$, 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

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

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 $\Delta_\infty^{Sk} - \Delta_\infty$, 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 $\Delta_\infty^{Sk=}$, 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 $\Delta_\infty^{Sk=}$, 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 $\Delta_\infty^{Sk=}$ 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. $\Delta_\infty^{Sk=}$ must be

consistent, because Δ_∞^{Sk} was consistent, and $\Delta_\infty^{Sk=}$ 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 $\Delta_\infty^{Sk=}$ 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 $m = \langle \Gamma, S_0, TELL, ASK \rangle$, we construct an m-model $M = \langle D, I, \gamma \rangle$ that satisfies all formulas in $\Delta_\infty^{Sk=}$, including φ itself. The domain D is the set $\{\Pi_1, \Pi_2, \ldots\}$ of equivalence classes of terms that occur in $\Delta_\infty^{Sk=}$. 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 $\pi(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \Delta_\infty^{Sk=}$, 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

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

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

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

for some $1 \leqslant j \leqslant n_i$, 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

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

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 $\leqslant i$, 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. □

5. Some properties of the logic

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 [10] 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

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

There are machines that satisfy this constraint; they are all limited in some other way, because no machine can be complete for first-order logic. For example, since provability in propositional logic is decidable, there can be machines that satisfy this constraint but don't accept sentences containing quantifiers.

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

$$ASK(S, \varphi \wedge \neg\varphi) = no \tag{14}$$

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 (14) and the constraint

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

requires more involved computation.

There are in fact machines that satisfy (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:

$$DECIDE(S, \neg\varphi) = 1 - DECIDE(S, \varphi),$$
$$DECIDE(S, \varphi \wedge \psi) = \min(DECIDE(S, \varphi), DECIDE(S, \psi)),$$
$$DECIDE(S, \varphi \vee \psi) = \max(DECIDE(S, \varphi), DECIDE(S, \psi)),$$
$$DECIDE(S, \varphi \supset \psi) = \max(1 - DECIDE(S, \varphi), DECIDE(S, \psi)).$$

If *ASK* is defined as

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

then (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. 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*. Unfortunately, this intuition cannot be realized using the semantics we have introduced in this paper. The problem is that in order to perform introspection, an agent's belief machine must be able to distinguish terms that refer to that agent from terms that don't. Since a belief machine simply manipulates syntactic objects, and has no information about their denotation, this is not possible.

A belief machine could be designed to treat one constant, say the constant *me*, differently from all others. When queried about a sentence $B(me, \varphi)$, it would perform an introspective self-query, but when queried about a sentence $B(\alpha, \varphi)$ for any α other than *me*, it would simply use its ordinary query-answering methods. This scheme would require some additional semantic apparatus beyond what we have introduced in this paper, because the denotation of the special constant *me* must not be allowed to vary from model to model.

It should be an indexical constant that, when it occurs outside of belief contexts, denotes the reasoner that is using it. Its denotation must also depend on context: in $B(a, P(me))$, it should have the same denotation as a, but in $B(a, B(b, P(me)))$ it should have the same denotation as b. This is so that $B(a, B(b, P(me)))$ entails $\exists x[x = b \wedge B(a, B(b, P(x)))]$. [3]

Given the indexical constant *me*, 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

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

There are non-trivial machines that satisfy the positive introspection constraint as well as constraints (C1)–(C4). 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)–(C4), 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,

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

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.

Axiom **5** is the negative introspection axiom. Using the special constant *me*, it can be written as

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

This axiom is made valid by the semantic constraint

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

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 subquery was *no*. However, unlike for the positive introspection constraint, only machines whose introspection is trivial can satisfy both this constraint and the finite basis constraint. It is a consequence of the finite basis constraint that any belief held in state S_0 must also

[3] Rapaport, Shapiro, and Wiebe [21] argue against the use of such a constant, because it complicates the semantics, and because it invalidates the axiom commonly used to define knowledge as true belief: the schema $K(\alpha, \varphi) \supset B(\alpha, \varphi) \wedge \varphi$ is not reasonable in a logic with an indexical term because, for example, from the premise $K(Fred, Rich(me))$, meaning "Fred knows he is rich", it licenses the conclusion $Rich(me)$, meaning "I am rich". We acknowledge these facts, but find the complication of the semantics a reasonable price to pay for the elegant way of handling introspection, and simulative inference about introspection, that an indexical constant allows; and the axiom of veridicality can be made acceptable simply by stipulating that *me* must not occur at the top level in φ (i.e., it occurs only inside an embedded belief context, if at all).

be held in every other state. If there is some sentence φ that is not believed in state S_0, an introspective machine must believe $\neg B(me, \varphi)$ in S_0. But then the finite basis constraint requires that $\neg B(me, \varphi)$ be believed in *every* state, even ones in which φ is also believed.

Since a machine with non-trivial negative introspection must violate the finite basis constraint, the soundness proof we gave for the rule of simulative inference does not apply; and, in fact, negative introspection can make simulative inference unsound. Let m be a simple machine that answers *yes* to sentences it has explicitly been *TELL*ed and *no* to all others, unless the query is of the form $\neg B(me, \varphi)$, in which case it performs a self-query on the sentence φ. For this machine,

$$ASK(TELL(S_0, P(c)), Q(d)) = no,$$

so from the premise $B(a, P(c))$, the rule of simulative inference licenses the conclusion $B(a, \neg B(me, Q(d)))$. Let $S = TELL(S_0, P(c), Q(d))$. Then

$$ASK(S, P(c)) = yes,$$

and

$$ASK(S, \neg B(me, Q(d))) = no.$$

Let M be an m-model such that the agent $|a|^M$ is assigned S as its belief state. Then the premise $B(a, P(c))$ is true in M, but the conclusion $B(a, \neg B(me, Q(d)))$ is not.

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

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

In [13], we show that, unlike the simulative inference rule, the rule of negative simulative inference *is* sound for some machines with negative introspection. Furthermore, we show that if the machine has both negative introspection and negative faithfulness (the converse of negative introspection), then the combination of the negative simulative inference rule and a negative introspection rule ($\frac{\neg B(\alpha,\varphi)}{B(\alpha,\neg B(me,\varphi))}$) can be used to derive positive conclusions of the kind yielded by the original simulative inference rule.

Note that negative introspective agents cannot exist in the deduction model, since there is no way to write a deduction rule whose premise is the *absence* of a belief. Konolige's chapter on introspection [17, Chapter 5] 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 [17, 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.

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 $\forall v\varphi$. The constraint

if $ASK(S, \varphi_{\tau/v}) = yes$ for every term τ,

then $ASK(S, \forall v\varphi) = yes$ for any variable v

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

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 $\varphi_{\tau/v}$, 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 $\varphi_{\tau'/v}$ for every ground term τ' it has seen in its *TELL*-history, and if all answers are *yes*, answering *yes* to $\varphi_{\tau/v}$, else *no*.
2. *ASK* answers $\forall v\varphi$ by *ASK*ing itself $\varphi_{\kappa/v}$ for some constant κ that has not appeared in its *TELL*-history, and answering *yes* if the answer to $\varphi_{\kappa/v}$ 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.[4] Now observe that if $\forall v B(\alpha, \varphi)$ holds and the domain of individuals is larger than the number of ground terms that have appeared in the agent's *TELL*-inputs, then $ASK(S, \varphi_{\tau/v})$ 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) $ASK(S, \varphi_{\tau'/v}) = yes$ for all ground terms τ', and so by (2) $ASK(S, \forall v\varphi) = yes$ as well, verifying the Barcan formula.

The converse Barcan formula,

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

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 $\forall v\varphi$. A sentence of the form $\forall v B(\alpha, \varphi)$, where v 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 $\varphi_{\tau/v}$. If $B(\alpha, \forall v\varphi)$ is satisfied by some model M, then it is also satisfied by a model identical to

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

M except that its domain contains one extra individual which is not denoted by any term. Such a model does not satisfy $\forall v\, B(\alpha, \varphi)$.

The constraint

if $ASK(S, \forall v\varphi) = yes$ for some variable v

then $ASK(S, \varphi_{\tau/v}) = yes$ 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

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

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 $ASK(S, \varphi_{\tau/v}) = yes$, then $ASK(S, \exists v\varphi) = yes$

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

6. Related work

We introduced the computational model of belief and simulative inference in [16]. The soundness theorem for simulative inference proved in the current paper is stronger than the one in [16], since it uses a weaker set of constraints (the combination of the robustness and commutativity constraints used in [16] has been replaced by a different commutativity constraint).

In an earlier version of this paper [15], the negative simulative inference rule was proved sound only for belief machines that satisfy a particular constraint; here, we have shown that that constraint follows from constraints (C1)–(C3). In [15] we also gave a proof of incompleteness for a broad class of belief machines. That proof is flawed, and the theorem is false.

There have been many proposals for modeling belief as limited reasoning, as a more realistic alternative to the possible worlds model's logical omniscience. Many of these proposals, e.g., the logic of Levesque [20], subsequently refined by Lakemeyer [19] and by Delgrande [5], and the logic of Fagin, Halpern, and Vardi [6], can be seen as making believers perfect reasoners in a weak logic in which entailment is decidable, rather than in full first-order logic. Our model is essentially compatible with this approach, but more general. Rather than choosing a single entailment or derivability relation to define believers' reasoning ability, we leave the reasoning method as a parameter of the model. The main results we have reached depend on the reasoning method being constrained in certain ways, but these constraints are weak enough that the results apply to an interesting range of different mechanisms.

Halpern, Moses, and Vardi [8] describe a model of "algorithmic knowledge" that is similar to our model in that an agent's beliefs are taken to be the sentences that the agent's

reasoning 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., there is no discussion of how an agent comes to be in a particular state, or how learning a particular new sentence would cause its 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 [7] is also simulative reasoning, but of a different kind. It can be summarized as follows: if α believes φ and has begun trying to decide whether ψ is true, and in a simulation of α's reasoning we are able to prove ψ from φ, then α believes ψ. This technique is sound for a broader class of reasoning mechanisms than ours: the closure constraint is not required for its soundness. The tradeoff is that for Haas' technique to be applicable, more information is required, namely the knowledge that α has wondered about ψ.

Our contribution has been a more precise understanding of the circumstances in which simulative reasoning is sound. There are many situations in which it is not sound, but is still desirable to use as a defeasible rule of thumb—for example, when the reasoning mechanisms of the believer and the observer are only similar, not identical. In such cases, the technique described by Chalupsky and Shapiro [2], treating simulative reasoning as a default inference rule, is applicable.

Ballim and Wilks [1] contend that there can be no satisfactory logical semantics of belief. 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. Our work incorporates computational mechanisms into reasoning about belief, without abandoning the logicist paradigm.

The work described here has been aimed at establishing the soundness of simulative reasoning given an arbitrary belief mechanism. In a related project, we are working on adding simulative reasoning to a particular mechanism, namely the EPILOG system [11, 22]. We are using the theoretical results described here to identify features of the reasoning mechanism that can cause simulative reasoning to yield incorrect conclusions. Also, while in this paper we have shown that internal details of the belief mechanism need not be considered when establishing the *soundness* of simulative reasoning, in our implementation work we have found that certain features of the mechanism's internal representations are critical to the potential *efficiency* of simulative reasoning. A preliminary report on these problems can be found in [12]; a more detailed account will be included in [14].

6.1. Konolige's deduction model

Our model of belief is similar to Konolige's deduction model [17]. 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.

6.1.1. 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.

While the belief machine abstraction allows agents that reason nonmonotonically, simulative inference is not sound for some of them. Whether a machine satisfies constraints (C1)–(C3) depends on the circumstances in which it retracts beliefs. A machine that chooses one or more *TELL*ed sentences to discard when it discovers that its inputs are contradictory can still satisfy the constraints, as demonstrated by the example machine in Section 3.2. In contrast, a machine that draws unsound conclusions from what it has been *TELL*ed, and later retracts those conclusions if new *TELL*ed sentences contradict them, cannot simultaneously satisfy the finite basis constraint (C2) and the commutativity constraint (C3). For this kind of machine, the form of simulative reasoning presented in this paper is not sound. However, in [13] we present another form that is sound under a weaker set of constraints, a set that does permit belief machines that reason defeasibly.

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.,

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

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 monotonically 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 [17, 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 10 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 10. *There exists a belief machine m, satisfying constraints* (C1)–(C4) *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. 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 the 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. Actually, 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 $j \geqslant 0$. The chosen tautologies are defined to have no terms other than variables as a simple way of satisfying the constraint of monotonicity under substitution (C4).

We define the output of *ASK* for database $\{\varphi_1, \ldots, \varphi_n\}$ $(n \geqslant 0)$ 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*.
- 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)–(C4). 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.

For the finite basis constraint (C2), it is clear from the definition of *TELL* and *ASK* that the database of a belief machine provides a set of formulas that are monotonically 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 commutativity constraint (C3) is obviously satisfied, since all *TELL*ed sentences are accepted (in any order), and the resultant state is independent of order of presentation.

The constraint of monotonicity under substitution (C4) 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 11. *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 $m = \langle \Gamma, S_0, TELL, ASK \rangle$, 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 $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$ (where $\mathbb{N}$ 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

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

and

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

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 monotonically acceptable sequence, and the credulity constraint, which says that every sequence is monotonically 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).

6.1.2. 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 [17, 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:

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

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 [17, 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.

6.1.3. Completeness in the deduction model

In Section 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 [17, 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 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.

6.1.4. 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 $B(a, P(\kappa))$ 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.

7. Conclusion

The entailments regarding belief in Kripke-style possible worlds models are too strong: an agent in such models must believe all of the consequences of its beliefs, no matter how much reasoning it would take to discover those consequences. This is an idealization that no real reasoner can satisfy. On the other hand, models of belief that use a more fine-grained theory of propositions, such as that of Cresswell [4] or Delgrande [5], license little in the way of reasoning about belief. In such models, for instance, it does not follow from the fact that someone believes $\varphi \wedge \psi$ that he also believes φ. There have been many proposals attempting to find a middle ground, where belief in a proposition entails belief in some of its obvious consequences, but not all of its consequences. Most such proposals essentially give a syntactic or semantic characterization of the consequences that follow obviously from a set of premises. Our approach is more general. We acknowledge that obviousness is not simply a property of sentences or propositions, but is related to the mechanism by which the believer accesses its beliefs. Therefore, our model of belief includes a model of the believer's storage and retrieval mechanism.

Simulative reasoning, an intuitive and potentially efficient technique for reasoning about the beliefs of other agents, is quite naturally expressed as an inference rule in our logic. Depending on the algorithm chosen to describe believers' reasoning abilities, the inference rule may or may not be sound. We have demonstrated that if the algorithm satisfies three relatively natural constraints, then the rule is sound. Given certain reasoning algorithms, no complete set of inference rules can exist for the logic, but we have given a set of rules that is complete for a syntactically restricted subset.

The approach taken in Konolige's deduction model [17] is similar, but in that model, believers are limited to a particular kind of reasoning, namely the exhaustive application of deductive inference rules. Our underlying model is fundamentally more general: it allows

reasoning to be the execution of an arbitrary algorithm. Simulative reasoning is not sound for every choice of algorithm, but we have shown that it is sound if the algorithm satisfies certain constraints. Even if we limit consideration to machines that satisfy these constraints, there are reasoners in the computational model that have no counterparts in the deduction model. These are reasoners that can choose to discard some of what they have learned if they discover that what they have learned is inconsistent. In the other direction, there are deduction structures that have no counterparts in the computational model, but they are purely abstract entities: they depend on an agent being able either to learn infinitely many facts explicitly, or to compute the uncomputable closure of a set of inference rules.

Expressiveness is not the only criterion for comparing the two models. If one wants to use a formal model of belief to justify the use of simulative reasoning in an actual system, then one needs to demonstrate that that particular system can be described using the vocabulary of the model. To apply the deduction model, one must be able to describe the reasoning performed by the system as the exhaustive application of a set of deductive inference rules. Unless the system happens to be implemented in that way, this may be difficult. It is more straightforward, in general, to show that a system's input/output behavior can be described using the belief machine abstraction, and then to show that the belief machine satisfies constraints (C1)–(C3).

Acknowledgements

The authors would like to thank James Allen, David Braun, Joseph Halpern, and the anonymous referee for their comments and suggestions. This work was supported by NSF research grants IRI-9623665 and IRI-9503312.

References

[1] A. Ballim, Y. Wilks, Artificial Believers (The Ascription of Belief), Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
[2] H. Chalupsky, S.C. Shapiro, Reasoning about incomplete agents, in: Proc. 5th International Conference on User Modeling, 1996.
[3] L.G. Creary, Propositional attitudes: Fregean representation and simulative reasoning, in: Proc. IJCAI-79, Tokyo, Japan, Vol. 1, 1979.
[4] M.J. Cresswell, Structured Meanings: The Semantics of Propositional Attitudes, MIT Press, Cambridge, MA, 1985.
[5] J.P. Delgrande, A framework for logics of explicit belief, Computational Intelligence 11 (1) (1995) 47–88.
[6] R. Fagin, J.Y. Halpern, M.Y. Vardi, A nonstandard approach to the logical omniscience problem, Artificial Intelligence 79 (2) (1995) 203–240.
[7] A.R. Haas, A syntactic theory of belief and action, Artificial Intelligence 28 (1986) 245–292.
[8] J.Y. Halpern, Y. Moses, M.Y. Vardi, Algorithmic knowledge, in: R. Fagin (Ed.), Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference, Morgan Kaufmann, San Francisco, CA, 1994, pp. 255–266.
[9] W. Hodges, Elementary predicate logic, in: D. Gabbay, F. Guenthner (Eds.), Elements of Classical Logic, Handbook of Philosophical Logic, Vol. I, Chapter 1, D. Reidel, Boston, MA, 1983.
[10] G.E. Hughes, M.J. Cresswell, An Introduction to Modal Logic, Methuen, London, 1968.
[11] C.H. Hwang, L.K. Schubert, Episodic logic: A comprehensive, natural representation for language understanding, Minds and Machines (Special Issue on KR in NLP) 3 (4) (1993) 381–419.

[12] A. Kaplan, Reason maintenance in a hybrid reasoning system, in: Workshop Proceedings, Inference in Computational Semantics, Amsterdam, August 1999. Revised version to appear in Journal of Language and Computation.
[13] A.N. Kaplan, Simulative inference about nonmonotonic reasoners, in: Theoretical Aspects of Rationality and Knowledge: Proceedings of the Seventh Conference, Morgan Kaufmann, San Mateo, CA, 1998, pp. 71–81.
[14] A.N. Kaplan, A computational model of belief, Ph.D. Thesis, University of Rochester, Rochester, NY, 2000.
[15] A.N. Kaplan, L.K. Schubert, Simulative inference in a computational model of belief, Technical Report 636, University of Rochester, Computer Science Department, 1997.
[16] A.N. Kaplan, L.K. Schubert, Simulative inference in a computational model of belief, in: H. Bunt, R. Muskens (Eds.), Computing Meaning, Studies in Linguistics and Philosophy, Kluwer, Dordrecht, 1999, pp. 185–202.
[17] K. Konolige, A Deduction Model of Belief, Morgan Kaufmann, Los Altos, CA, 1986.
[18] S.A. Kripke, Semantical considerations on modal logic, Acta Philosophica Fennica 16 (1963) 83–94.
[19] G. Lakemeyer, Limited reasoning in first-order knowledge bases, Artificial Intelligence 71 (1994) 213–255.
[20] H.J. Levesque, A logic of implicit and explicit belief, in: Proc. AAAI-84, Austin, TX, 1984, pp. 198–202.
[21] W.J. Rapaport, S.C. Shapiro, J.M. Wiebe, Quasi-indexicals and knowledge reports, Cognitive Sci. 21 (1) (1997) 63–107.
[22] S. Schaeffer, C.H. Hwang, J. de Haan, L.K. Schubert, The Users's Guide to EPILOG, Boeing Co., Edmonton, Alberta, 1991.