Rule-based and Resource-bounded: A New Look at Epistemic Logic*

Mark Jago

Abstract

Syntactic logics do not suffer from the problems of logical omniscience but are often thought to lack interesting properties relating to epistemic notions. By focusing on the case of rule-based agents, I develop a framework for modelling resource-bounded agents and show that the resulting models have a number of interesting properties.

1 Introduction

Logical omniscience is a well-documented problem for epistemic logics based on a possible worlds semantics (first presented in Hintikka's seminal *Knowledge and Belief* [21]). In this paper, I concentrate on the concept of belief, as believing ϕ is a necessary condition on knowing ϕ. Belief is defined as truth in all epistemically accessible worlds and as a consequence, belief is closed under consequence and agents automatically believe all valid sentences. This is clearly inadmissible as a general analysis of belief.[1] Several authors take the view that, in a number of situations, logical omniscience is unproblematic, "in particular for interpretations of knowledge that are often appropriate for analyzing distributed systems ... and certain AI systems." However, "it is certainly not appropriate to the extent that we want to model resource-bounded agents" [16, p. 41]. I will therefore take as my starting point the requirement that the beliefs of resource-bounded agents be modelled accurately.

To avoid the problem of logical omniscience, a syntactic approach is required: that is, one which takes the truth-conditions of belief ascriptions to be given, at least in part, in terms of sentences.[2] Contrary to the impression one receives from the logical literature, syntactic accounts of belief receive support from the current philosophical literature.[3] An objection is that syntactic epistemic logics merely give us "ways of *representing* knowledge [and belief] rather than *modelling* knowledge [and belief]". If so, the thought runs, "[o]ne gains very little intuition about knowledge [or belief] from studying syntactic structures" [15, p. 320]. The syntactic approach "lacks the elegance and intuitive appeal of the semantic [possible worlds] approach" [14, p. 40]. My aim in this paper is therefore to present an elegant and intuitively appealing syntactic logic of belief which allows us to accurately model resource-bounded reasoners.

The key idea is to model inference as a nondeterministic step-by-step process. Each time an inference rule is applied and a new belief derived, the agent moves into a new belief state. This is a very fine-grained notion of belief change. It allows models to be built in which perfectly rational reasoning is possible, in the sense that the agent's logical abilities need not be depleted in any way, but in which logical omniscience never arises. This framework models agents that, as [22] has it, are neither logically omniscient nor logically ignorant. The lesson to be taken is that, in order to model real AI agents without making unrealistic assumptions about their resource bounds, an epistemic logic must be able to represent an agent's reasoning at the level of individual inferences (the title of the paper is intended to reinforce this point). My strategy in this paper is to investigate step-by-step inference in a simplified setting. The only inferential action that will be modelled here is the act of deriving new beliefs from old using (a generalised version of) *modus ponens*.

*Thanks to Natasha Alechina for incisive comments, to Brian Logan for guidance and to three anonymous referees for the Logics for Resource Bounded Agents workshop for their helpful suggestions.

[1] See [34, 35, 24] for discussions of logical omniscience and related problems.

[2] Many authors seem to dispute this claim, especially [27, 14, 16, 15]. However, none of the approaches presented there genuinely solve the problem. See [24]. The approach based on *awareness* given in [14] unwittingly concedes the point (see [24].

[3] See Perry [30, 31] and Corazza [11, 10] for accounts of belief in terms of an accepted sentence. Further support comes from accepting the *language of thought hypothesis*: see Fodor [17, 18].

Actions such as making assumptions or instantiating axiom schema are not modelled here (but see [24] in which such actions are modelled in the current framework).

I take as a working example a prominent case from the AI literature: the case of *rule-based agents*. These agents consist of a program—a set of condition-action rules—and a rule interpreter. Rule-based agents have been more or less ignored by the literature on epistemic logic[4] but play an important rôle in other areas of AI. There are several rule-based agent architectures available, e.g. SOAR [26] and SIM-AGENT [33] which allow a great degree of abstraction in specifying behaviour. Rule-based programming extensions are also increasingly being offered as add-ons to existing, lower-level, agent toolkits, e.g., JADE [7] and FIPA-OS [32]. Rule-based behaviour is also playing an important rôle in analysing domains such as business. Business rules (statements that define or constrain an aspect of a business [9], e.g. *every visitor of the conference gets a 20 per cent discount on the first product purchase*) are being used by companies to analyse the behaviour and improve the efficiency of their business. As the business rules community puts it, "business rules are the very essence of a business. They define the terms and state the core business policies. They control or influence business behaviour. They state what is possible and desirable in running a business—and what is not" [9].

In general, a rule-based agent's program will contain condition-action rules of the form

$$\mathtt{P}_1, \ldots, \mathtt{P}_n \Rightarrow \mathtt{Q}_1, \ldots, \mathtt{Q}_m$$

$\mathtt{P}_i$ are the conditions, $\mathtt{Q}_i$ the resulting actions, and each $\mathtt{P}_i$, $\mathtt{Q}_i$ may contain unbound variables or possibly even logical connectives.[5] Here, I treat both rules in the agent's program and literals held in its working memory as beliefs (the working memory does not play a significant rôle in the formalism). In the modal systems discussed below, an agent's rules are represented in the states of those models. An equivalent formulation could be given by encoding rules as conditions on the arcs between states. Intuitively, it makes sense to encode *inference* rules as conditions on arcs and beliefs as the sentences supported by states (the question is whether to treat the rules that appear in the agent's program as inference rules). On the alternative formulation, each rule is treated as an inference rule in its own right whereas on the account presented here, rules are formulae and the agent reasons using a generalised form of *modus ponens*:

$$\frac{\lambda_1, \ldots, \lambda_n, (\lambda_1, \ldots, \lambda_n \Rightarrow \lambda)}{\lambda}$$

In this way, agents are modelled as having many beliefs and only the one rule of inference.

I focus on an agent's reasoning process by assuming that the agent has an initial stock of beliefs (which might be observations) that are neither revised nor added to, other than by firing rules and adding their consequents as new beliefs. I make three further simplifying restrictions: (i) to rules which produce a single action; (ii) to propositional rules and (iii) to rules which contain no disjunctions (thus, on agents who have no disjunctive beliefs). The first two are inessential;[6] (iii) *is* a restriction on the expressiveness of the logic presented here, but is by no means a limitation of the general framework.[7]

The remainder of the paper proceeds as follows. In section 2, I present syntax and semantics for a logic which models a single rule-based agent and then, in section 3, discuss the properties of such models. In section 4, I consider an agent with a fixed program and, in section 5, present an axiomatization and complexity analysis of the resulting logic. Related and future work is discussed in sections 6 and 7.

2 Modelling Rule-Based Agents

We fix a denumerable set of propositions $\mathcal{P} = \{p_1, p_2 \ldots\}$. A literal is either a proposition or its negation; literals are written $\lambda_1, \lambda_2 \ldots$. Rules are of the form $\lambda_1, \ldots, \lambda_n \Rightarrow \lambda$ and in general rules are denoted $\rho, \rho_1, \rho_2, \ldots$. Since it is often useful to know which belief a rule adds when fired, we use the abbreviation

[4] A notable exception is [25]; see section 6.

[5] For example, in the 'definition' rule $\mathtt{person}(x) \Rightarrow \mathtt{man}(x) \vee \mathtt{woman}(x)$.

[6] Because negation may only appear before a predicate—the agent does not believe the negation of any rule—a program in which rules contain unbound variables can be modelled using a denumerable set of propositions, so long as both the set of predicates and the set of constants is denumerable (in any practical case, both will be finite). Using a propositional logic allows us to use a far more readable notation without limiting the underlying logic—all results given below also hold for the predicate case. A logic that deals with predicate-style rules is considered in [4].

[7] Disjunction is ignored here merely to reduce the complexity of the presentation. See [24] for the extended framework, including disjunctions.

$\mathsf{cn}(\rho)$ for λ, given that $\rho = (\lambda_1, \ldots, \lambda_n \Rightarrow \lambda)$. The agent's *internal language* $\mathcal{L}^{\mathcal{P}}$ over $\mathcal{P}$ contains only rules and literals; no other formulae are considered well-formed. Since $\mathcal{P}$ will be fixed throughout, the superscript may be informally dropped. Arbitrary formulae of $\mathcal{L}$ are denoted $\alpha, \alpha_1, \ldots$.

The modal language $\mathcal{ML}^{\mathcal{P}}$, which is used to reason about the agent's beliefs, is built from formulae of $\mathcal{L}^{\mathcal{P}}$ (again the superscript may informally be dropped). $\mathcal{ML}$ contains the usual propositional connectives $\neg, \wedge, \vee, \rightarrow$, the '$\Diamond$' modality and a belief operator B. Given a literal λ and a rule ρ, '$\mathsf{B}\lambda$' and '$\mathsf{B}\rho$' are primitive wffs of $\mathcal{ML}$, and all primitive wffs are formed in this way. If ϕ_1 and ϕ_2 are both $\mathcal{ML}$ wffs, the complex wffs of $\mathcal{ML}$ are then given by

$$\neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \Diamond\phi_1$$

The dual modality '$\Box$' is introduced by definition: $\Box\phi \stackrel{df}{=} \neg\Diamond\neg\phi$. Note that the primitive formulae of $\mathcal{ML}$ are all of the form '$\mathsf{B}\alpha$', where 'α' is a $\mathcal{L}$-formula, hence the problem of substitution within belief contexts does not arise in logics based on $\mathcal{ML}$.

Models are graphs of states, with each arc representing a change in an agent's belief state. Although time is not explicitly represented in these models, each arc is thought of as a transition from an agent's belief state at one time to a (possible) belief state at a future moment in time, arrived at by firing a rule and adding its consequent as a new belief. A model M is a structure $\langle S, T, V\rangle$ where S is a set of states; $T \subseteq S \times S$ is a transition relation on states; and $V : S \longrightarrow 2^{\mathcal{L}}$ is the *labelling function*, assigning a set of sentences of the agent's internal language to each state. Where there is a transition from s to s', s' will be said to be a *successor* of s; s' is *reachable* from s when there is a sequence of states $ss_1s_2\cdots s_ns'$ such that each is the successor of the one before.

Definition 1 (Labelling) *Given a model $M = \langle S, T, V\rangle$, a sentence $\alpha \in \mathcal{L}$ is said to* label *a state $s \in S$ when $\alpha \in V(s)$. Given models $M = \langle S, T, V\rangle$ and $M' = \langle S', T', V'\rangle$ (which need not be distinct), states $s \in S$ and $s' \in S'$ are said to be label identical, written $s \leftrightsquigarrow_{\mathcal{L}} s'$, when $V(s) = V'(s')$.*

The definition of a formula ϕ of $\mathcal{ML}$ being true, or satisfied, by state s in a model M (written $M, s \Vdash \phi$) is as follows:

$M, s \Vdash \mathsf{B}\alpha$ iff $\alpha \in V(s)$

$M, s \Vdash \neg\phi$ iff $M, s \nVdash \phi$

$M, s \Vdash \phi_1 \wedge \phi_2$ iff $M, s \Vdash \phi_1$ and $M, s \Vdash \phi_2$

$M, s \Vdash \phi_1 \vee \phi_2$ iff $M, s \Vdash \phi_1$ or $M, s \Vdash \phi_2$

$M, s \Vdash \phi_1 \rightarrow \phi_2$ iff $M, s \nVdash \phi_1$ or $M, s \Vdash \phi_2$

$M, s \Vdash \Diamond\phi$ iff there exists a state $s' \in S$ such that Tss' and $M, s' \Vdash \phi$

Such models are known as Kripke models. '$M, s \Vdash \phi$' is read as *s supports the truth of ϕ in M*, or *s supports ϕ* for short (if it is clear which model is being talked about). The definitions of global satisfiability and validity are standard, and these notion extend to sets for formulae in the usual way. States $s, s' \in S$ are said to be *modally equivalent* in M, written $s \leftrightsquigarrow s'$, when $\{\phi \mid M, s \Vdash \phi\} = \{\psi \mid M, s' \Vdash \psi\}$.

Because these models are common to modal logics in general, they need to be restricted in certain ways to model rule-based agents. In particular, the rules which an agent believes do not change; rules are neither learnt nor forgot. This is standard practise in rule-based AI systems (*cf* condition $\mathsf{S4}$ below). Secondly, T must relate states s and u when some rule ρ can be fired at s, and u is just like s except the agent has gained one new belief, the consequent of ρ. Here, ρ is said to be an s-matching rule.

Definition 2 (Matching rule) *Let ρ be a rule of the form $\lambda_1, \ldots, \lambda_n \Rightarrow \lambda$. ρ is then said to be s-matching, for some state $s \in S$, iff $\rho \in V(s)$, each $\lambda_1, \ldots, \lambda_n \in V(s)$ but $\lambda \notin V(s)$.*

Whenever ρ is s-matching for some state s, then the agent can move into a new state u in which it has gained a new belief. u is said to *extend* s by that new belief, namely $\mathsf{cn}(\rho)$.

Definition 3 (Extension of a state) *For any rule ρ and states $s, u \in S$, u extends s by $\mathsf{cn}(\rho)$ iff $V(u) = V(s) \cup \{\mathsf{cn}(\rho)\}$.*

If there are no matching rules at a state (and so no rule instances to fire), that state is a *terminating state* and has a transition to itself (or to another identical state, which amounts to much the same in modal logic). This ensures that every state has an outgoing transition; in other words, T is a *serial* relation. As a consequence, the question 'what will the agent be doing after n cycles?' may always be answered, even if the agent ran out of rules to fire in less than n cycles.

Definition 4 (Terminating state) *A state s is said to be a terminating state in a model M iff no rule ρ is s-matching.*

Transitions relate terminating states to identically labelled terminating states and, whenever there is a matching rule ρ at a state s, a transition should only be possible to a state u which extends s by $\mathsf{cn}(\rho)$. We capture such transition systems in the class **S** (for single agent models).

Definition 5 *The class **S** contains precisely those models M which satisfy the following:*

S1 *for all states $s \in S$, if a rule $\lambda_1, \ldots, \lambda_n \Rightarrow \lambda$ is s-matching, then there is a state $s' \in S$ such that Tss' and s' extends s by λ.*

S2 *for any terminating state $s \in S$, there exists a state $s' \in S$ such that $V(s') = V(s)$ and Tss'*

S3 *for all states $s, s' \in S$, Tss' only if either* (i) *there is an s-matching rule $\lambda_1, \ldots, \lambda_n \Rightarrow \lambda$ and s' extends s by λ; or* (ii) *s is a terminating state and $V(s) = V(s')$.*

S4 *for all rules ρ and states $s, u \in S$, $\rho \in V(s)$ iff $\rho \in V(u)$.*

It is clear that this definition ensures that T is a serial relation for any model $M \in \mathbf{S}$. For any state $s \in S$, either there is at least one matching rule or there is not. In the former case, S1 ensures that s is related to some extension of itself by T; otherwise, s is a terminating state and is related to an identically labelled state by T.

There may, of course, be many matching rules at a given state, and for each there must be a state u such that Tsu. Each transition may be thought of as corresponding to the agent's nondeterministic choice to fire one of these rule instances. '$\Diamond\phi$' may then be read as 'after some such choice, ϕ will hold.' We can think of the agent's reasoning as a cycle: (i) match rules against literals; (ii) choose one matching rule; (iii) add the consequent of that rule to the set of beliefs; repeat. By chaining diamonds (or boxes), e.g. '$\Diamond\Diamond\Diamond$' we can express what properties can (and what will) hold after so many such cycles. We can abbreviate sequences of n diamonds (or n boxes) as $\Diamond^n$ and $\Box^n$ respectively. '$\Box^n\phi$', for example, may be read as 'ϕ is guaranteed to hold after n cycles.' Note that the agent's set of beliefs grows monotonically state by state and that the agent never revises its beliefs, even if they are internally inconsistent.

Example

Before investigating the properties that models in the class **S** have, an example may help to illustrate the concepts that have been introduced. Typically, the rules in rule-based programs will contain variables which are matched against the contents of the agent's working memory to produce instances of the rule. In this example, the agent's program contains just two rules:

R1 $\mathtt{PremiumCustomer}(x), \mathtt{Product}(y) \Rightarrow \mathtt{Discount}(x, y, 10\%)$

R2 $\mathtt{Spending}(x, > 1000) \Rightarrow \mathtt{PremiumCustomer}(x)$

However, a first-order language is not needed to model this agent. Instead, we can consider the language that contains all instances of the rules and all ground literals that appear in these instances.[8]

Now suppose that the agent's initial working memory contains the beliefs

$$\mathtt{Product}(\mathtt{iBook}) \qquad \mathtt{Spending}(\mathtt{Jones}, > 1000) \qquad \mathtt{Product}(\mathtt{Sunglasses})$$

When the agent begins executing, R2 can be matched against Jones to produce

$$\mathtt{Spending}(\mathtt{Jones}, > 1000) \Rightarrow \mathtt{PremiumCustomer}(\mathtt{Jones}) \tag{1}$$

[8] When considering a program $\mathcal{R}$ (section 4) or the axiomatization given in section 5, we must also assume that the set of constants used to instantiate the variables in rules is finite.

Since no other instances of either R1 or R2 are possible, there is then a unique next state in which

$$\texttt{PremiumCustomer}(\texttt{Jones})$$

is added to the agent's working memory. At the agent's next cycle, R1 can be matched against Jones and either Sunglasses or iBook to produce the instances

$$\texttt{PremiumCustomer}(\texttt{Jones}), \texttt{Product}(\texttt{Sunglasses}) \Rightarrow \texttt{Discount}(\texttt{Jones}, \texttt{Sunglasses}, 10\%) \quad (2)$$

$$\texttt{PremiumCustomer}(\texttt{Jones}), \texttt{Product}(\texttt{iBook}) \Rightarrow \texttt{Discount}(\texttt{Jones}, \texttt{iBook}, 10\%) \quad (3)$$

Note that (1) is no longer counted as a matching rule instance, since its consequent has already been added to the working memory. The agent can then move into a state in which the working memory contains either $\texttt{Discount}(\texttt{Jones}, \texttt{Sunglasses}, 10\%)$ or else contains $\texttt{Discount}(\texttt{Jones}, \texttt{iBook}, 10\%)$ in addition to its previous contents. If the agent fires (2), adding $\texttt{Discount}(\texttt{Jones}, \texttt{Sunglasses}, 10\%)$ to working memory, (3) remains a matching rule instance and $\texttt{Discount}(\texttt{Jones}, \texttt{iBook}, 10\%)$ is added at the next state. Similarly, if the agents fires (3), adding $\texttt{Product}(\texttt{iBook}) \Rightarrow \texttt{Discount}(\texttt{Jones}, \texttt{iBook}, 10\%)$, then (2) remains matching. There is then a next state adding $\texttt{Discount}(\texttt{Jones}, \texttt{Sunglasses}, 10\%)$ to working memory. Figure 1 shows a branching time model in which new beliefs are added to the working memory (only new beliefs are shown). The agent can derive $Discount(Jones, iBook, 10\%)$ in 2 cycles, whereas it must derive it in 3 cycles. If this model is M and its root s, then $M, s \Vdash \Diamond\Diamond\texttt{Discount}(\texttt{Jones}, \texttt{iBook}, 10\%)$ and $M, s \Vdash \Box\Box\Box\texttt{Discount}(\texttt{Jones}, \texttt{iBook}, 10\%)$.

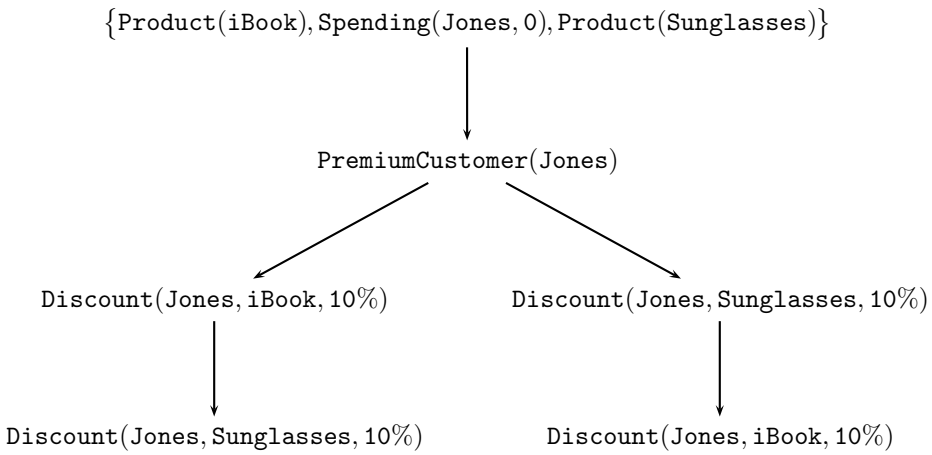

Figure 1: New literals added to WM

3 Properties of Models

Now we need to know, how well do these models capture a rule-based agent's reasoning process? Below I give a number of simple yet powerful results. Firstly, there is a strong relationship between the way states are labelled, the modal formulae which hold at those states and bisimulation. Secondly, models have a *belief convergence* property. The remainder of section is fairly technical.

When a bisimulation relation Z holds between states s, s', we write $s \backsimeq s'$.[9] Intuitively, all bisimilar models describe the same reasoning process. It is sometimes convenient to work with models in which the transition relation T forms a tree on the states S. Such models are known as *tree models.*

Proposition 1 *Some standard properties of models $M = \langle S, T, V\rangle$ and $M' = \langle S', T', V'\rangle$:*

a. For all $s \in S$ and $s' \in S'$, $s \backsimeq s'$ implies $s \leftrightsquigarrow s'$. [8, p.67]

b. Every model M has a bisimilar tree model (obtained by unravelling M).

c. Any satisfiable formula ϕ of depth d is satisfiable in a tree model of height no greater than d.

[9] See, for example, [8] for an explanation of bisimulation.

d. (Hennessy-Milner Theorem) If M and M' are image finite,[10] *then, $s \leftrightarrows s'$ implies $s \trianglelefteq s'$ for all $s \in S$ and $s' \in S$. [8, p.69]*

These are standard properties of all Kripke models. From (a) and (b), whenever we are working with a model M, we can always switch to a tree model M' which satisfies the same formulae (if $M, s \Vdash \phi$, then there is a state $s' \in M' : M', s' \Vdash \phi$.) The converse to (b) does not hold in general.[11] (c) gives a restricted version of the converse to (b). I now list a few properties which models $M \in \mathbf{S}$ in particular possess (I don't give a proof here as each proof is more or less immediate).

Proposition 2 (Properties of S Models) *Assume that a model $M = \langle S, V, T\rangle$ is a tree model with root r. Then:*

a. For all states s, s' of depth n, $|V(s)| = |V(s')|$. If $V(r)$ is finite and s, s' are not terminating states, then $|V(s)| = n + |V(r)|$.

b. If $V(r)$ is finite, then $V(s)$ is finite for all $s \in S$.

c. If $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$ and s, s' are not terminating states, then s and s' are of the same depth.

d. All siblings of terminating nodes are also terminating nodes.

e. If two children s_1 and s_2 of s are such that $V(s_1) - \{\lambda_1\} = V(s_2) - \{\lambda_2\}$ then each has a child s' such that $V(s') = V(s) \cup \{\lambda_1, \lambda_2\}$.

Lemma 1 *For tree models $M, M' \in \mathbf{S}$ and states s in M, s' in M': if $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$ and Tsu, then there is a $u' \in S'$ such that $Ts'u'$ and $u \leftrightarrows\!\!\!\!\!\!\sim\; u'$.*

Proof: If s is a terminating state then this is trivial; so assume that this is not the case. Then there is an s-matching rule ρ such that $V(u) = V(s)\{\mathsf{cn}(\rho)\}$. Since $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$, ρ is also s'-matching, hence there is a u' such that $Ts'u'$ and $V(u') = V(s') \cup \{\mathsf{cn}(\rho)\}$; hence $u \leftrightarrows\!\!\!\!\!\!\sim\; u'$. ⊣

Theorem 1 *For any models $M, M' \in \mathbf{S}$ and all states s in M and s' in M': $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$ iff $s \leftrightarrows s'$.*

Proof: Clearly, $s \leftrightarrows s'$ implies $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$. The converse: $M, s \Vdash \phi$ iff $M', s' \Vdash \phi$, whenever $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$, is shown by induction on the complexity of ϕ. The base case is trivial so assume that $M, v \Vdash \psi$ iff $M', v' \Vdash \psi$ for all $v \in S$, $v' \in S'$ and ψ of lower complexity than ϕ whenever $v \leftrightarrows\!\!\!\!\!\!\sim\; v'$. The cases for Booleans are also trivial, so consider $\phi := \Diamond\psi$. Then $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$ and $M, s \Vdash \Diamond\psi$ implies that there is a state $u \in S$ such that Tsu and $M, u \Vdash \psi$. By lemma 1, there is a state $u' \in S'$ such that $Ts'u'$ $u' \leftrightarrows\!\!\!\!\!\!\sim\; s'$. By hypothesis, $M', u' \Vdash \psi$ and hence $M', s' \Vdash \Diamond\phi$. The converse holds by a similar argument, hence $s \leftrightarrows s'$. ⊣

Theorem 2 *For any models $M, M' \in \mathbf{S}$ and all states s in M and s' in M': $s \leftrightarrows s'$ iff $s \trianglelefteq s'$.*

Proof: From proposition 1(a), $s \trianglelefteq s'$ implies $s \leftrightarrows s'$, so it only remains to show the converse. Assume $s \leftrightarrows s'$ and that there is a $u \in S$ such that Tsu; we must show that there is a state $u' \in S'$ such that $T's'u'$ and $u \leftrightarrows u'$. If s is a terminating state, this is trivial; so assume that s is non-terminating. There must be an s-matching rule ρ. Since $s' \leftrightarrows\!\!\!\!\!\!\sim\; s$, ρ must also be s'-matching and so, by $\mathsf{S1}$, there is a state $u' \in S'$ such that $T's'u'$ extending s' by $\mathsf{cn}(\rho)$. Hence $u' \leftrightarrows\!\!\!\!\!\!\sim\; s'$ and so, by theorem 1, $u' \leftrightarrows s'$. ⊣

Corollary 1 *Let $M = \langle S, T, V\rangle \in \mathbf{S}$. For any $s, s' \in S$ and any descendant u of s, if $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$ then there is a descendant u' of u such that $u \leftrightarrows\!\!\!\!\!\!\sim\; u'$.*

Proof: The proof is immediate from theorem 2. ⊣

We can thus partition states into equivalence classes ($s, s' \in [s]$ whenever $s \leftrightarrows\!\!\!\!\!\!\sim\; s'$) and transform any model M into a bisimular model $M^{\equiv}$ just by comparing the labels on states in M. The domain of $M^{\equiv}$ is the set of label equivalence classes in M such that $T^{\equiv}[s][u]$ whenever Tsu and $V^{\equiv}([s]) = V(s)$ for some $s \in [s]$. Any formula satisfiable in M is then satisfiable in $M^{\equiv}$, and $M^{\equiv}$ has the handy property that $[s] \leftrightarrows\!\!\!\!\!\!\sim\; [u]$ implies $[s] = [u]$.

[10] A model is image finite iff $\bigcup_{s\in S}\{u \mid Tsu\}$ is finite.

[11] Given a model M, we can construct a modally equivalent model N containing an infinite branch for which there can be no bisimulation $Z : M \trianglelefteq N$ (if we suppose there is, we will eventually come to a point on the infinite branch in N for which the corresponding point in M has no successor; hence they cannot be bisimilar states).

Definition 6 *Let $M = \langle S, T, V\rangle \in \mathbf{S}$ and $n \in \mathbb{N}$. Define $T^n su$ to hold iff there are states $s_0 \cdots s_n$ such that $s = s_0, u = s_n$ and, for each $i < n$, $Ts_n s_{n+1}$.*

Now we show that models in $\mathbf{S}$ have the property of *belief convergence*.

Theorem 3 (Belief Convergence) *For any model $M = \langle S, T, V\rangle \in \mathbf{S}$, any state $r \in S$ and any $n \in \mathbb{N}$, if $T^n rs$ and $T^n ru$, then there is a state s' reachable from s and u' reachable from u such that $s' \leftrightsquigarrow_{\mathcal{L}} u'$.*

Proof: Without loss of generality, consider a tree model $M \in \mathbf{S}$ whose root is r. Let s, u both be reachable from r in a finite number of transitions. Then there are equinumerous sets X, Y such that $V(s) = V(r) \cup X$ and $V(u) = V(r) \cup Y$. Now consider the subbranch from r to s: for each transition Tvv' on the branch, pick a v-matching rule ρ such that v' extends v by $\mathsf{cn}(\rho)$. Enumerate the selected rules ρ for which $\mathsf{cn}(\rho) \notin V(u)$ as $\rho_1, \ldots, \rho_n$ (from r to s). It is easy to see that there must be a state u' reachable from u, on the branch that results from firing first ρ_1 and then ... and then ρ_n. Thus $V(u') = V(u) \cup \{\mathsf{cn}(\rho_1), \ldots, \mathsf{cn}(\rho_n)\} = V(u) \cup X' = V(u) \cup X = V(r) \cup Y \cup X$. By similar reasoning, there must be a state s' reachable from s with $V(s') = V(s) \cup Y = V(r) \cup X \cup Y$. Hence, $s' \leftrightsquigarrow_{\mathcal{L}} u'$. ⊣

4 Finite Models and Programs

Because of our motivating interest in resource boundedness, we will sometimes want to restrict ourselves to models in which each state is labelled by only finitely many $\mathcal{L}$-formulae, for these are the sentences representing the agent's basic explicit beliefs, of which any real agent may have only finitely many at any one time. We capture this intuition in the class of *finite memory models*.

Definition 7 (Finite memory model) *A model $M \in \mathbf{S}$ is a finite memory model iff $V(s)$ is finite for each $s \in S$. $\mathbf{C}^{fm}$ is the set of all finite memory models in some class $\mathbf{C}$.*

An interesting feature of finite memory models in $\mathbf{S}$ is that each is bisimilar to a finite state model in $\mathbf{S}$. This is the *finite model property*:

Theorem 4 (Finite Model Property) *For any finite memory model $M = \langle S, T, V\rangle \in \mathbf{S}^{fm}$, there is a model M' containing only finitely many states and a bisimulation $Z : M \backsimeq M'$.*

Proof: For any state $s \in S$, if $V(s)$ is finite, s may only have finitely many children, each of which are labelled by only finitely many formulae. Let R be the set of rules which label each state (by S4, all states are labelled by precisely the same rules); clearly R is finite. Then any state $s \in S$ can have at most $|\{\mathsf{cn}(\rho) \mid \rho \in R\}|$ matching rules. Thus a finite memory model with infinitely many states must have an infinite branch, on which only a finite initial segment is generated by matching rules, i.e. only the first n states on the branch are non-terminating states, for some $n \leq |\{\mathsf{cn}(\rho) \mid \rho \in R\}|$. By S3ii, $s \leftrightsquigarrow_{\mathcal{L}} s'$ whenever Tss' and s, s' are terminating states. A model M' can be obtained by selecting the first terminating state s on each branch in M, removing all the descendants of s and adding a transition Tss. M' satisfies S2 and is clearly bisimilar to M. Moreover, since s occurred on a finite initial segment of a branch in M, M' only contains branches of finite length. It follows that M' only contains finitely many states. ⊣

The above has been a general characterisation of rule-based agents which execute fixed but unspecified set of rules. However, we are often interested in restricting our attention to agents reasoning with a specific set of rules. Following the usual terminology, a *program* is simply a finite set of rules. One of the uses of the current approach is testing for properties of particular programs.[12] Given a program $\mathcal{R}$ for the agent, we can define a subclass $\mathbf{S}_{\mathcal{R}}$ as containing just those models in $\mathbf{S}$ in which the agent believes all the rules in $\mathcal{R}$ and no further rules.

Definition 8 (The class $\mathbf{S}_{\mathcal{R}}$) *Let $\mathcal{R}$ be a program (i.e. a finite set of rules). A model $M = \langle S, T, V\rangle \in \mathbf{S}_{\mathcal{R}}$ iff $M \in \mathbf{S}$ and, for all states $s \in S$, $\mathcal{R} \subseteq V(s)$. An $\mathcal{L}$-formula ϕ is said to be $\mathbf{S}_{\mathcal{R}}$-satisfiable iff it is satisfied at some state s in some model $M \in \mathbf{S}_{\mathcal{R}}$.*

[12] In [24] I discuss adding additional temporal operators and path quantifiers from *computational tree logic* (CTL), a common input language for model checking technology. This extension allows rule-based programmers to use current model checking technology to verify their programs.

Each class $\mathbf{S}_\mathcal{R}$ is a subclass of $\mathbf{S}$ and each model in $\mathbf{S}$ is in exactly one class $\mathbf{S}_\mathcal{R}$. $\mathbf{S}$ and its subclasses differ with respect to (semantic) entailment and satisfiability. If $\mathcal{R} = \{p \Rightarrow q\}$, then $\mathsf{B}p \wedge \neg\Diamond\mathsf{B}q$ is $\mathbf{S}$-satisfiable but not $\mathbf{S}_\mathcal{R}$-satisfiable; similarly, $\Diamond\mathsf{B}q$ is a $\mathbf{S}_\mathcal{R}$-consequence but not a $\mathbf{S}$-consequence of $\mathsf{B}p$. The remainder of this section surveys some properties of the class $\mathbf{S}_\mathcal{R}$, including a decidability result.

Theorem 5 *Let $\mathcal{R}$ be a program, ϕ be any $\mathcal{ML}$ formula and $n = |\{\,\mathsf{cn}(\rho) \mid \rho \in \mathcal{R}\}|$. If ϕ is $\mathbf{S}_\mathcal{R}$-satisfiable at all, then it is satisfiable in a finite model $M \in \mathbf{S}_\mathcal{R}$ containing at most n^n states.*

Proof: Suppose ϕ is satisfiable at s in a model in $\mathbf{S}_\mathcal{R}$; then it is satisfied by a tree model $M \in \mathbf{S}_\mathcal{R}$ whose root is s (proposition 1). By S3, any state u in M can have at most $|\mathcal{R}|$ children. Now, take any state s in M of depth n. No $\rho \in \mathcal{R}$ can be s-matching, for otherwise, some ancestor of s must have extended its parent by some $\lambda \notin \{\,\mathsf{cn}(\rho) \mid \rho \in \mathcal{R}\}$; but S3 prohibits this. Then any state at depth n or greater must be a terminating state. There is then a model $M' \in \mathbf{S}_\mathcal{R}$ forming a rooted directed acyclic graph, bisimilar to M, in which $s \leftrightsquigarrow u$ implies $s = u$ (e.g. by taking equivalence classes from M, as described above). For any state s in M', $|\{s' \mid Tss'\}| \leq n$ and, for states u, u' at depth n or greater, $T'uu'$ implies $u = u'$. Therefore M' can contain at most n^n states. ⊣

In any state in a model $M \in \mathbf{S}_\mathcal{R}$, only the labels in the sets $\mathcal{R}$ and $\{\lambda_1, \ldots, \lambda_n, \lambda \mid (\lambda_1, \ldots, \lambda_n \Rightarrow \lambda) \in \mathcal{R}\}$ can have any effect on which rules do and do not match at that state. Thus, it is only these formulae that affect the structure that T forms on S. Labels that are not from these sets may be removed without changing which states are accessible from which in the model. We can combine this with standard techniques to get a notion of filtration for $\mathbf{S}_\mathcal{R}$ models.

Definition 9 ($\mathcal{R}$-filtration) *Let Γ be closed under both subformulae and negation; and set*

$$L_\Gamma \;=\; \mathcal{R} \cup \{\alpha \mid \mathsf{B}\alpha \in \Gamma\} \cup \{\lambda_1, \ldots, \lambda_n, \lambda \mid (\lambda_1, \ldots, \lambda_n \Rightarrow \lambda) \in \mathcal{R}\}$$

An $\mathcal{R}$-filtration of $M = \langle S, T, V\rangle$ through Γ is then a model $M_\Gamma = \langle S, T, V_\Gamma\rangle$ where $V_\Gamma(s) = V(s) \cap L_\Gamma$.

Filtration here is rather different than in regular modal logic. Here, we must ensure that rules and the beliefs needed for them to match are not removed from states when we filter, hence the use of L_Γ.

Lemma 2 *Let Γ be as above, $M = \langle S, T, V\rangle \in \mathbf{S}_\mathcal{R}$ and M_Γ be the $\mathcal{R}$-filtration of M through Γ. Then for any $\phi \in \Gamma$ and $s \in S$: $M, s \Vdash \phi$ iff $M_\Gamma, s \Vdash \phi$.*

Proof: By induction on the complexity of ϕ. If ϕ is an $\mathcal{ML}$ primitive this is trivial. So assume that, for all $\psi \in \Gamma$ of complexity $k < n$ and any state $s \in S$: $M, s \Vdash \psi$ iff $M^L, s \Vdash \psi$. We show this holds for all $\phi \in \Gamma$ of complexity n. The *only if* direction is trivial; in the *if* direction, consider these cases:

$\phi := \neg\psi$. Then $M, s \nVdash \psi$ and, by hypothesis, $M_\Gamma, s \nVdash \psi$, hence $M_\Gamma, s \Vdash \phi$.

$\phi := \psi_1 \wedge \psi_2$. Then $M, s \Vdash \psi_1$ and $M, s \Vdash \psi_2$. By hypothesis, $M_\Gamma, s \Vdash \psi_1$ and $M_\Gamma, s \Vdash \psi_2$, hence $M_\Gamma, s \Vdash \phi$.

$\phi := \Diamond\psi$. Then there is a $s' \in S$ such that $M, s' \Vdash \psi$ and Tss'. By hypothesis, $M_\Gamma, s' \Vdash \psi$ and hence $M_\Gamma, s \Vdash \phi$.

The other Boolean cases are similar; it follows that $M_\Gamma, s \Vdash \phi$. ⊣

Lemma 3 *Let Γ be as above, $M \in \mathbf{S}_\mathcal{R}$ and M_Γ be the $\mathcal{R}$-filtration of M through Γ. Then $M_\Gamma \in \mathbf{S}_\mathcal{R}$.*

Proof: It follows from lemma 2 that any rule ρ is s-matching in M iff it is s-matching in M_Γ and that $\rho \in V_\Gamma(s)$ iff $\rho \in V(s)$. Since T is common to both M and M_Γ, S1-4 are satisfied and hence $M_\Gamma \in \mathbf{S}_\mathcal{R}$.⊣

Definition 10 *Let $sub(\phi)$ be the set of subformulae of ϕ, i.e.:*

$sub(\mathsf{B}\alpha) = \{\mathsf{B}\alpha\}$

$sub(\neg\phi) = sub(\Diamond\phi) = sub(\phi)$

$sub(\phi \wedge \psi) = sub(\phi \vee \psi) = sub(\phi \rightarrow \psi) = sub(\phi) \cup sub(\psi)$

and let $Cl(\phi)$ be $sub(\phi)$ closed under negation.

Theorem 6 (Finite Memory Property) *Let $\mathcal{R}$ be a program and ϕ be any $\mathcal{ML}$ formula. If ϕ is $\mathcal{R}$-satisfiable, then it is satisfiable in a finite memory model $M \in \mathbf{S}_{\mathcal{R}}^{fm}$.*

Proof: Assume that M, s satisfies ϕ. Let M_Γ be the $\mathcal{R}$-filtration of M through $\Gamma = Cl(\phi)$. By lemma 2, $M_\Gamma, s \Vdash \phi$ and, by lemma 3, $M_\Gamma \in \mathbf{S}_{\mathcal{R}}$. Since $Cl(\phi)$ and $\mathcal{R}$ are both finite, $V(s)$ is finite for every $s \in S$, hence $M_\Gamma \in \mathbf{S}_{\mathcal{R}}^{fm}$. $\dashv$

Theorem 7 (Decidability) *Let $\mathcal{R}$ be a program and ϕ be any $\mathcal{ML}$ formula. Then it is decidable whether ϕ is $\mathbf{S}_{\mathcal{R}}$-satisfiable.*

Proof: Suppose ϕ is $\mathcal{R}$-satisfiable; then it is satisfied at the root r of some tree model $M \in \mathbf{S}_{\mathcal{R}}$. Let M_Γ be the $\mathcal{R}$-filtration of M through $\Gamma = Cl(\phi)$. By inspecting the proof of theorem 6, $M_\Gamma, r \Vdash \phi$, $M_\Gamma \in \mathbf{S}_{\mathcal{R}}^{fm}$ and $V_\Gamma(r) = V(s) \cap L_\Gamma$, with L_Γ as definition 9. Let $n = |\{\, \mathsf{cn}(\rho) \mid \rho \in \mathcal{R}\}|$. By inspecting the proof of theorem 5, a model M'_Γ can be obtained that has at most n^n states (e.g. by taking equivalence classes from M_Γ, as described above). Thus if ϕ has an $\mathbf{S}$-model, one can be found by considering each model with no more then n^n states whose root is labelled by a subse t of L_Γ. Since L_Γ is bounded by the size of ϕ and $\mathcal{R}$, we have an upper bound on the search for a model. We therefore have a terminating algorithm that will find an $\mathbf{S}_{\mathcal{R}}$ model for ϕ if one exists. $\dashv$

5 Axiomatization and Complexity

Given some such program $\mathcal{R}$, it is easy to axiomatize the logic of the class $\mathbf{S}_{\mathcal{R}}$. The abbreviation

$$\mathsf{match}(\lambda_1, \ldots, \lambda_n \Rightarrow \lambda) \stackrel{df}{=} \mathsf{B}\lambda_1 \wedge \cdots \wedge \mathsf{B}\lambda_n \wedge \neg\mathsf{B}\lambda$$

is helpful. The axiom system shown in figure 2 is called $\Lambda_{\mathcal{R}}$. **A6** says that, when a belief is added, it must have been is added by some matching rule instance in $\mathcal{R}$. **A7** says that, if all matching rule instances in the current state are $\rho_1, \ldots, \rho_n$, then each of the successor states should contain the consequent of one of those instances.

Cl all classical propositional tautologies

K $\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$

A1 $\mathsf{B}\rho$ where $\rho \in \mathcal{R}$

A2 $\neg\mathsf{B}\rho$ where $\rho \notin \mathcal{R}$

A3 $\mathsf{B}\alpha \rightarrow \Box\mathsf{B}\alpha$

A4 $\mathsf{B}(\lambda_1, \ldots, \lambda_n \Rightarrow \lambda) \wedge \mathsf{B}\lambda_1 \wedge \cdots \wedge \mathsf{B}\lambda_n \Rightarrow \Diamond\mathsf{B}\lambda$

A5 $\Diamond(\mathsf{B}\alpha \wedge \mathsf{B}\beta) \rightarrow \mathsf{B}\alpha \vee \mathsf{B}\beta$

A6 $\Diamond\mathsf{B}\alpha \rightarrow (\mathsf{B}\alpha \vee \bigvee_{\lambda_1,\ldots,\lambda_n \Rightarrow \lambda \in \mathcal{R}, \lambda = \alpha} \mathsf{B}\lambda_1 \wedge \cdots \wedge \mathsf{B}\lambda_n)$

A7 $\mathsf{match}\rho_1 \wedge \cdots \wedge \mathsf{match}\rho_n \wedge \bigwedge_{\rho \neq \rho_{i \leq n}, \rho \in \mathcal{R}} \neg\mathsf{match}\rho \rightarrow \Box\Big(\mathsf{B}\,\mathsf{cn}(\rho_1) \vee \cdots \vee \mathsf{B}\,\mathsf{cn}(\rho_n)\Big)$ $n > 1$

A8 $\Diamond\top$

MP $\dfrac{\phi \;\; \phi \rightarrow \psi}{\psi}$

N $\dfrac{\phi}{\Box\phi}$

Figure 2: Axiom schemes for $\Lambda_{\mathcal{R}}$

A *derivation* in $\Lambda_{\mathcal{R}}$ is defined in a standard way, relative to $\mathcal{R}$: ϕ is derivable from a set of formulae Γ (written $\Gamma \vdash_{\mathcal{R}} \phi$) iff there is a sequence of formulae $\phi_1, \ldots, \phi_n$ where $\phi_n = \phi$ and each ϕ_i is either an instance of an axiom schema, or a member of Γ, or is obtained from the preceding formulae by **MP** or **N**. Suppose an agent's program $\mathcal{R}$ contains the rules $\rho_1, \ldots, \rho_n$. This agent is guaranteed to reach a state in which it believes α in k steps, starting from a state where it believes $\lambda_1, \ldots, \lambda_m$, iff

$$\mathsf{B}\rho_1 \wedge \ldots \wedge \mathsf{B}\rho_n \wedge \mathsf{B}\lambda_1 \wedge \ldots \wedge \mathsf{B}\lambda_m \Rightarrow \Box^k \mathsf{B}\alpha$$

is derivable in $\Lambda_{\mathcal{R}}$ (again, $\Box^k\alpha$ is an abbreviation for $\Box\Box\cdots\Box\alpha$, k times). We now show that $\Lambda_{\mathcal{R}}$ is the logic of the class $\mathbf{S}_{\mathcal{R}}$ (the proofs of lemmas 4 and 5 are standard).

Lemma 4 (Lindenbaum lemma) *Any set of formulae Γ can be expanded to a $\Lambda_{\mathcal{R}}$-maximal consistent set Γ^+.*

A canonical model $M^{\mathcal{R}} = \langle S, T, V\rangle$ is built in the usual way. States in S are $\Lambda_{\mathcal{R}}$-maximal consistent sets; Tsu iff $\{\phi \mid \Box\phi \in s\} \subseteq u$ (or equivalently, iff $\{\Diamond\phi \mid \phi \in u\} \subseteq s$). Finally, $V(s) = \{\alpha \in \mathcal{L} \mid \mathsf{B}\alpha \in s\}$, for each $s \in S$.

Lemma 5 (Existence and Truth lemma) *For any ϕ and any state s in $M^{\mathcal{R}}$: (i) if there is a formula $\Diamond\phi \in s$ then there is a state u in $M^{\mathcal{R}}$ such that Tsu and $\phi \in u$; and (ii) $M^{\mathcal{R}}, s \Vdash \phi$ iff $\phi \in s$.*

Lemma 6 *Let $M^{\mathcal{R}}$ be a canonical model and let $\alpha \in \mathcal{L}$ and $s, u \in S$. Then (i) if Tsu and $\alpha \in V(u)$ but $\alpha \notin V(s)$, then $V(u) = V(s) \cup \{\alpha\}$; and (ii) α in part (i) must be a literal.*

Proof: Part (i) follows from the definition of '$\Vdash$' together with the truth lemma and the fact that states are closed under axioms **A3** and **A5**. The former axiom ensures that s is a subset of u, the latter ensures that α is the only new belief. For part (ii), if we suppose α were some rule we would have $\alpha \in \mathcal{R}$ and so $\alpha \in s$, contrary to hypothesis. $\dashv$

Theorem 8 (Completeness) *$\Lambda_{\mathcal{R}}$ is strongly complete with respect to the class $\mathbf{S}_{\mathcal{R}}$: given a program $\mathcal{R}$, a set of $\mathcal{ML}$-formulae Γ and an $\mathcal{ML}$-formula ϕ, $\Gamma \Vdash_{\mathcal{R}} \phi$ only if $\Gamma \vdash_{\mathcal{R}} \phi$.*

Proof: Expand Γ to a $\Lambda_{\mathcal{R}}$-maximal consistent set Γ^+ from which we build a canonical model $M^{\mathcal{R}}$. From the truth lemma, it follows that $M^{\mathcal{R}}, \Gamma^+ \Vdash \Gamma$. It remains only to show that $M^{\mathcal{R}}$ is in the class $\mathbf{S}_{\mathcal{R}}$, i.e. that $M^{\mathcal{R}}$ satisfies S1–S4. S4 is clearly satisfied; the remaining cases are:

$M^{\mathcal{R}}$ satisfies S1: Assume there is an s-matching rule ρ. Given the truth lemma, it is easy to see that each of its antecedents is a member of s, whereas its consequent is not. **A4** and the existence lemma guarantee an accessible state u which, given lemma 6, is the extension of s by $\mathsf{cn}(\rho)$.

$M^{\mathcal{R}}$ satisfies S2: Suppose s is a terminating state. By axiom **A8**, there is an accessible state s'. By axiom **A6**, $\alpha \in V(s')$ implies $\alpha \in V(s)$ for any literal α (this holds because there are no matching rules at s). It then follows from axioms **A1**–**A3** that $V(s') = V(s)$, hence S2 is satisfied.

$M^{\mathcal{R}}$ satisfies S3: Suppose Tsu for states s, u in $M^{\mathcal{R}}$. By definition, $\{\phi \mid \Box\phi \in s\} \subseteq u$. By axiom **A7**, there must be one literal believed in u but not in s, namely the consequent of either ρ_1 or $\ldots$ or ρ_n. Then by the argument just used, it follows that u is the extension of s by this new belief. $\dashv$

Theorem 9 *Given a particular program $\mathcal{R}$, the problem of deciding whether a formula ϕ is satisfiable in a model $M \in \mathbf{S}_{\mathcal{R}}$ is NP-complete.*

Proof: Clearly the problem is NP-hard. Let $n = |\{\mathsf{cn}(\rho) \mid \rho \in \mathcal{R}\}|$. From theorem 5, any $\mathbf{S}_{\mathcal{R}}$-satisfiable sentence ϕ has a tree model $M \in \mathbf{S}_{\mathcal{R}}$ containing no more than n^n states which, given the proof of theorem 6, is no larger than $|\phi|n^n$. Given any Kripke structure M', state s in M' and a modal formula ψ, it takes time polynomial in the size of M' and ψ to check whether $M', s \Vdash \psi$ [8]. The crucial point here is that $|\mathcal{R}|$, and hence n^n, is constant in $\mathbf{S}_{\mathcal{R}}$. Thus, we can guess a model $M \in \mathbf{S}_{\mathcal{R}}$ of size no greater than $|\phi|n^n$ and check whether ϕ is satisfied at the root of M in time polynomial in $|\phi|$. It follows that the problem of deciding whether ϕ is $\mathbf{S}_{\mathcal{R}}$-satisfiable is in NP. $\dashv$

One of the main practical uses of models in a class $\mathbf{S}_{\mathcal{R}}$ is to check whether $\mathcal{R}$ satisfies certain properties, specified as an input formula ϕ. One may want to check a range of different programs against a different property: for example, suppose a developer requires an agent which can never move into a state in which ϕ holds. On discovering that ϕ is $\mathbf{S}_{\mathcal{R}_1}$-satisfiable, she must reject $\mathcal{R}_1$. If $\mathcal{R}_2$ is the next generation of the program, then ϕ needs to be checked for $\mathbf{S}_{\mathcal{R}_2}$-satisfiability. The evolution from $\mathcal{R}_1$ to $\mathcal{R}_2$ may have added a large number of rules to the program. This example highlights that it is not just the scalability of satisfiability checking given ϕ as an input that should concern us. How the problem scales with the size of the agent's program is also crucial.[13] An interesting problem to consider, therefore, is the one that takes *both* a formula ϕ and a program $\mathcal{R}$ as its input and determines whether ϕ is $\mathbf{S}_{\mathcal{R}}$ satisfiable. I call this the **S**-SAT problem. The complexity of the problem should be investigated in terms of $|\mathcal{R}|$ and $|\phi|$ rather than in terms of $|\phi|$ alone.

Theorem 10 **S***-SAT is in PSPACE.*

Proof: The proof is similar to the proof that the K-satisfiability problem has a PSPACE-implementation in [8]. An **S**-Hintikka set over a program $\mathcal{R}$ and a set Σ is like a standard Hintikka set but, in addition, contains all instances of axiom schemes **A1**–**A8** over Σ. A *witness set* is then defined as in [8]. The key result is that a **S**-Hintikka set H over $\mathcal{R}$ and Σ is $\mathbf{S}_{\mathcal{R}}$-satisfiable iff there is a witness set generated by H over $\mathcal{R}$ and Σ. A formula ϕ can then be tested for satisfiability by setting $\Sigma = Cl(\phi) \cup \{\mathsf{B}\alpha \mid \alpha \in Cl(\mathcal{R})\}$. A correct algorithm called *witness* can then be given which returns *true* on input $H, \mathcal{R}, \Sigma$ iff H is a $\mathbf{S}_{\mathcal{R}}$-satisfiable **S**-Hintikka set over $\mathcal{R}$ and Σ. The final stage establishes that *witness* has an implementation on a non-deterministic Turing machine that only requires space polynomial in $|\phi|$ and $|\mathcal{R}|$. Since NPSPACE = PSPACE, this establishes that **S**-SAT is in PSPACE. The full proof is given in [24]. ⊣

6 Related Work

Early work in epistemic logic on rule-based system, influenced by work in AI, is found in Konolige's *Deduction Model of Belief* [25]. As here, semantics is given in terms of sets of formulae, with $\mathsf{B}_i\alpha$ true iff agent i has α in its belief set. Each agent i is assigned a set of deduction rules ρ_i, which need not be logically complete (and in fact must not be to avoid closure of belief under classical consequence). A belief set is then obtained by closing an agent's knowledge base under its rules. This is what [13] term a "final tray" model of belief (p. 1), reporting what an agent *would* derive, given unlimited time and memory. Agents with a functionally complete set of deduction rules are therefore modelled as believing all tautologies and all consequences of their beliefs and so logical omniscience is only avoided by considering agents with depleted logical ability.

In [22, 23], Ho Ngoc Duc presents an epistemic logic based on dynamic logic. If r is an inference rule that the agent can use, then $\langle r \rangle$ is the usual dynamic modality 'after executing (i.e. reasoning using) r, it is possible that ...' where the blank will usually be filled with a belief ascription. Ho introduces a future modality $\langle F \rangle$, defined as the iterated set of all choices of actions $r_1, \ldots, r_n$ available to the agent: $F = (r_1 \cup \cdots \cup r_n)^*$. $\langle F \rangle \mathsf{B}\phi$ then says that the agent can come to believe that ϕ and $[F]\mathsf{B}\phi$ says that the agent must believe that ϕ at some point in the future. The notion of the future here is thus an idealised one, considering all the states in a temporally unbounded reasoning process. For example, if p is a propositional (modality-free) tautology, then $\langle F \rangle \mathsf{B}p$ is a theorem. It is not even correct to read $\langle F \rangle \mathsf{B}p$ as 'the agent can believe p at some point in the (idealised) future' (just consider a tautology p so large that *no* agent could come to hold the sentence in its memory). The $\langle F \rangle$ operator thus ignores resource bounds.

This highlights an important point. Avoiding logical omniscience is not an end in itself. Evidently, what is therefore required is a logic which not only avoids logical omniscience, but that captures the *stages* of reasoning are captured, rather than just the idealised endpoint. Step logic [13] attempts to overcome this problem by indexing beliefs by time points or *steps*. Each step corresponds to a cycle in the agent's reasoning. Step logic deduction rules take the form:

$$\frac{t\text{:} \quad \alpha_1 \ \cdots \ \alpha_n}{t+1\text{:} \qquad \alpha}$$

[13] In fact, this can often be the more important factor of the two, for the size of many programs currently in use far exceeds the size of the formulae that it is useful to check for satisfiability.

However, a semantics is not provided for any step logic in [13]. A minimal possible worlds semantics for step logic are found in [29] and [12]. Belief is defined as a relation between a world and a set of sets of worlds, based on Scott-Montague (or neighbourhood/minimal) structures; an axiomatization is found in [29]. However, agents are modelled as believing all propositional tautologies and their beliefs as closed under equivalence. This is a limitation of Scott-Montague semantics, which deals with the *intensions* of believed sentences (equivalent sentences necessarily have identical intensions). Grant, Kraus and Perlis provide a first-order axiomatization and model theory for step logic in [20]. Not all of the models they describe are adequate representations of an agent's beliefs, in that a particular model may contain 'extra' sentences not derivable from the agent's previous beliefs. Accordingly, they introduce the notion of *knowledge supported models*. This suggests that the framework is not ideally suited to modelling belief obtained by rule-based reasoning.

Timed Reasoning Logic (TRL) is introduced in [5, 6]. The focus is on modelling different rule application and conflict resolution strategies in rule-based systems, building on the step logic approach. Semantics are provided in terms of syntactic *local models*. TRL uses labelled formulae rather than the modal metalanguage adopted here. In [36], TRL is used to model assumption-based reasoning in resource-bounded agents. Such ways of reasoning cannot be modelled by step logic, in which implications must be deal with by forming instances of Hilbert axioms. One major difference between TRL and the present approach is that an agent's current state together with its rules determines a unique next state. It is thus not possible to distinguish between the beliefs that an agent can derive from those it must derive in a certain number of steps. This is a limitation of TRL (and step logic) that has been addressed int he present work.

Ågotnes [1] considers a logic of *finite syntactic epistemic states*. As with TRL and the Deduction Model, the semantics is based on sets of sentences. An unusual feature of [1] is that syntactic operators take *sets* of sentences as their arguments. $\triangle_i\{\phi_1, \ldots, \phi_n\}$ says that agent i believes at least that $\phi_1, \ldots, \phi_n$ are true. Similarly, $\bigtriangledown_i\{\phi_1, \ldots, \phi_n\}$ says that agent i believes at the most that $\phi_1, \ldots, \phi_n$ are true. The syntax of what an agent believes at a time thus closely follows the semantics. A semantics is provided by game-theoretic structures, allowing expressive ATL modalities to be incorporated in the logic. Given a set of agents G, the path quantifier $\langle\!\langle G \rangle\!\rangle$ allows sentences to express co-operation between members of G to achieve some result. This approach forms the basis of [3] and [2].

7 Future Work

This paper has presented a basic framework for modelling rule-based agents in a simplified, monotonic setting. One of the principal applications of the logic that has been developed is to verify that a rule-based program satisfies certain criteria. To this end, the addition of computational tree logic (CTL) modalities would constitute an increase in expressivity and allow the resulting language to be used as an input to model checkers. Note that the $\Diamond$ modality discussed here corresponds to the CTL modality $E\mathcal{X}$ ($E\mathcal{X}\phi$ holds iff ϕ holds at the next step of some branch). This is a minor amendment to the syntax; the models themselves remain identical. The aim in this paper was explicitly to restrict attention to a single rule-based agent. As with most modal logics, it is surprisingly easy to add multiple agents to the formalism (add a valuation V_i for each agent i and plausible rules about communication between agents); see [4].

A more challenging development would be to drop the monotonicity requirement. Nonmonotonic reasoning is important in many areas of AI: see [19]. In fact, a good deal of practical reasoning is nonmonotonic. Makinson comments that "almost all of our everyday reasoning is nonmonotonic; purely deductive, monotonic inference takes place only in rather special contexts, notably pure mathematics" [28, p. 19]. Nonmonotonic reasoning in rule-based systems can arise in a number of ways. One is when certain conditions determine which rule should be fired in the next cycle. Situations can arise in which ρ may fire but, if the agent were to know more information, ρ would not be fired. The resulting consequence relation is nonmonotonic.

Another route to nonmonotonicity in rule-based systems is to consider rules of the form

$$\mathtt{P}_1, \ldots, \mathtt{P}_n \Rightarrow {\sim}\mathtt{Q}$$

where ${\sim}\mathtt{Q}$ instructs the agent to remove $\mathtt{Q}$ from its working memory. Firing such a rule does not lead to a new belief; but it can lead to the agent having one less belief. Amending the current framework to allow for such nonmonotonic rule-based inference would increase its applicability in many areas of AI. A starting point is to amend the requirement that one state extends another when there is a transition to the

first from the second. Instead, define an *amend* operation '$\circ$' on $2^{\mathcal{L}} \times \mathcal{L}$ such that $X \circ p = X \cup \{p\}$ and $X \circ {\sim}p = X - \{p\}$. Then, whenever there is an s-matching rule ρ, there is a state u such that Tsu and u *amends* s by $\mathsf{cn}(\rho)$. In this system, the order in which rules fire matters. Moreover, it is no longer the case that if Γ entails ϕ then $\Gamma \cup \{\psi\}$ entails ϕ. It would be interesting to see which of the properties discussed above hold of this logic; this is left for future work.

8 Summary

This paper presents a framework for modelling resource bounded reasoners that derive new beliefs from old through inference. The approach is designed to handle inference rules of many types. The example of rule-based programs was chosen here as it allows a simple testbed for the framework. The resulting models of rule-based agents have a number of interesting properties: the equivalence between label identity, modal equivalence and bisimulation, and the belief convergence property. When a particular program is specified, a logic with a decidable satisfaction relation is obtained, which can be easily axiomatized. The interesting satisfiability problem in the resulting logics is in PSPACE.

Not all reasoners are rule-based in the restricted sense used here. Many agents revise their beliefs (and indeed their rules); conclusions can be withdrawn as well as asserted; agents reason inductively and abductively as well as deductively; agents make assumptions and see what follows. These forms of reasoning have not been addressed here. Nevertheless, resource-bounded reasoners using any of these forms of reasoning will amend their set of beliefs in a step-by-step way according to their chosen set of rules. By treating these transitions from one belief state to the next as the foundation for a semantics, a fine-grained account of resource-bounded reasoning is possible in which the problems of logical omniscience never arise.

References

[1] Thomas Ågotnes. *A Logic of Finite Syntactic Epistemic States*. PhD thesis, Department of Informatics, University of Bergen, Norway, April 2004.

[2] Thomas Ågotnes and N. Alechina. The dynamics of syntactic knowledge. Technical Report 304, Dept. of Informatics, Univ. of Bergen, Norway, 2005.

[3] Thomas Ågotnes and M. Walicki. Syntactic knowledge: A logic of reasoning, communication and cooperation. In *Proceedings of the Second European Workshop on Multi-Agent Systems (EUMAS 2004)*, 2004.

[4] N. Alechina, M. Jago, and B. Logan. Modal logics for communicating rule-based agents. In *Proceedings of ECAI 06*, 2006.

[5] N. Alechina, B. Logan, and M. Whitsey. A complete and decidable logic for resource-bounded agents. In *Proc. Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*. ACM Press, July 2004.

[6] Natasha Alechina, Brian Logan, and Mark Whitsey. Modelling communicating agents in timed reasoning logics. In *proc. JELIA 04*, pages 95–107, Lisbon, September 2004.

[7] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software Practice and Experience*, 21(2):103–128, 2001.

[8] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, New York, 2002.

[9] Business rules community website, accessed 13-03-06. `http://www.brcommunity.com/`, 2006.

[10] Eros Corazza. *Reflecting the Mind: Indexicality and Quasi-Indexicality*. Oxford University Press, 2004.

[11] Eros Corazza. Singular propositions, quasi-singular propositions, and reports. In Kepa Korta, editor, *Semantics, Pragmatics, and Rhetoric*. CSLI, Stanford, 2004.

[12] J. Elgot-Drapkin, S. Kraus, M. Miller, M. Nirkhe, and D. Perlis. Active logics: A unified formal approach to episodic reasoning. Technical Report CS-TR-4072, University of Maryland, Department of Computer Science, 1999.

[13] J. Elgot-Drapkin, M. Miller, and D. Perlis. Memory, reason and time: the Step-Logic approach. In R. Cummins and J. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 79–103. MIT Press, Cambridge, Mass., 1991.

[14] R. Fagin and J.Y. Halpern. Belief, awareness and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[15] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT press, 1995.

[16] R. Fagin, J.Y. Halpern, and M.Y. Vardi. A nonstandard approach to the logical omniscience problem. In R. Parikh, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. Third Conference*, San Fransisco, California, 1990. Morgan Kaufmann.

[17] Jerry Fodor. *Psychosemantics: The Problem of Meaning in the Philosophy of Mind*. MIT Press, Cambridge, Mass., 1987.

[18] Jerry Fodor. *A Theory of Content and Other Essays*. MIT Press, Cambridge, Mass., 1990.

[19] M. Ginsberg. AI and nonmonotonic reasoning. In D.M. Gabbay *et al*, editor, *Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, pages 1–33. Clarendon Press, Oxford, 1994.

[20] John Grant, Sarit Kraus, and Donald Perlis. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems*, 2000.

[21] J. Hintikka. *Knowledge and belief: an introduction to the logic of the two notions*. Cornell University Press, Ithaca, N.Y., 1962.

[22] D.N. Ho. Logical omniscience vs. logical ignorance. In C.P. Pereira and N. Mamede, editors, *Proceedings of EPIA'95*, volume 990 of *LNAI*, pages 237–248. Springer, 1995.

[23] D.N. Ho. Reasoning about rational, but not logically omniscient, agents. *Journal of Logic and Computation*, 5:633–648, 1997.

[24] Mark Jago. *Logics for Resource-Bounded Agents*. PhD thesis, University of Nottingham, 2006. Forthcoming.

[25] K. Konolige. *A Deduction Model of Belief*. Morgan Kaufman, 1986.

[26] J. E. Laird, A. A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[27] H. J. Levesque. A logic of implicit and explicit belief. In *National Conference on Artificial Intelligence*, pages 1998–202, 1984.

[28] David Makinson. *Bridges from Classical to Nonmonotonic Logic*, volume 5 of *Texts in Computing*. King's College Publications, 2005.

[29] M. Nirkhe, S. Kraus, and D. Perlis. Thinking takes time: a modal active-logic for reasoning in time. Technical Report CS-TR-3249, University of Maryland, Department of Computer Science, 1994.

[30] John Perry. Belief and acceptance. *Midwest Studies in Philosophy*, 5:553–54, 1980.

[31] John Perry. *The Problem of the Essential Indexical*. Oxford University Press, Oxford, 1993.

[32] S. Poslad, P. Buckle, and R. G. Hadingham. The fipa-os agent platform: Open source for open standards. In *Proceedings of the Fifth International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (PAAM2000)*, pages 355–368, Manchester, April 2000.

[33] A. Sloman and B. Logan. Building cognitively rich agents using the sim agent toolkit. *Communications of the ACM*, 42(3):71–77, March 1999.

[34] R. Stalnaker. The problem of logical omniscience I. *Synthese*, 89:pp.425–440, 1991.

[35] M. Whitsey. Logical omniscience: a survey. Technical Report NOTTCS-WP-2003-2, School of Computer Science and IT, University of Nottingham, 2003.

[36] M. Whitsey. Timed reasoning logics: An example. In *Proc. of the Logic and Communication in Multi-Agent Systems workshop (LCMAS 2004)*. Loria, 2004.