Computationally grounded account of belief and awareness for AI agents

Natasha Alechina and Brian Logan

Abstract

We discuss the problem of designing a computationally grounded logic for reasoning about epistemic attitudes of AI agents, mainly concentrating on beliefs. We briefly review exisiting work and analyse problems with semantics for epistemic logic based on accessibility relations, including interpreted systems. We then make a case for syntactic epistemic logics and describe some applications of those logics in verifying AI agents.

1 Introduction

The *Belief-Desire-Intention* (BDI) model of agency is arguably the most widely adopted approach to modelling artificial intelligence agents [18]. In the BDI approach, agents are both characterised and programmed in terms of propositional attitudes such as beliefs and goals and the relationships between them. For the BDI model to be useful in developing AI agents, we must be able to correctly ascribe beliefs and other propositional attitudes to an agent. However standard epistemic logics suffer from several problems in ascribing beliefs to computational agents. Critically, it is not clear how to connect the computational implementation of an agent to the beliefs we ascribe to it. As a result, standard epistemic logics model agents as logically omniscient. The concept of logical omniscience was introduced by Hintikka in [19], and is usually defined as the agent knowing all logical tautologies and all the consequences of its knowledge. However, logical omniscience is problematic when attempting to build realistic models of agent behaviour, as closure under logical consequence implies that deliberation takes no time. For example, If processes within the agent such as belief revision, planning and problem solving are modelled as derivations in a logical language, such derivations require no investment of computational resources by the agent.

In this paper we present an alternative approach to modelling agents which addresses these problems. We distinguish between beliefs and reasoning abilities which we ascribe to the agent ('the agent's logic') and the logic we use to reason *about* the agent. In this we follow, e.g., [21, 20, 17]. In the spirit of [33], our logic to reason about the agent's beliefs is grounded in a concrete computational model. However, unlike [33, 29] we choose not to interpret the agent's beliefs as propositions corresponding to sets of possible states or runs of the agent's program, but syntactically, as formulas 'translating' some particular configuration of variables in the agent's internal

1

state. One of the consequences of this choice is that we avoid modelling the agent as logically omniscient. This has some similarities with the bounded-resources approach of [15] and more recent work such as [1, 4].

This paper is essentially a high-level summary of the course on logics and agent programming languages the authors gave at the 21st European Summer School in Logic, Language and Information held in Bordeaux in 2009. Some of the ideas have appeared in our previous work, for example [5, 6], but have never been summarised in a single article.

The rest of the paper is organised as follows. In section 2 we discuss motivations for modelling intentional attitudes of AI agents in logic. In section 3 we analyse problems with the standard semantics for epistemic logic, including interpreted systems. In section 4 we discuss other approaches to modelling knowledge and belief, namely the syntactic approach, logic of awareness, and algorithmic knowledge. Then we introduce our proposal based on the syntactic approach in section 5 and briefly survey some of the applications of the syntactic approach in verification of agent programs in section 6.

2 Logic for verification

There are many reasons for modelling agents in logic. The focus of our work is on specifying and verifying AI agents using logic. The specification and verification of agent architectures and programs is a key problem in agent research and development. Formal verification provides a degree of certainty regarding system behaviour which is difficult or impossible to obtain using conventional testing methodologies, particularly when applied to autonomous systems operating in open environments. For example, the use of appropriate specification and verification techniques can allow agent researchers to check that agent architectures and programming languages conform to general principles of rational agency, or agent developers to check that a particular agent program will achieve the agent's goals in a given range of environments.

Ideally, such techniques should allow specification of key aspects of the agent's architecture and program, and should admit a fully automated verification procedure. One such procedure is model-checking [12]. Model-checking involves representing the system to be verified as a transition system M which can serve as a model of some (usually temporal) logic, specifying a property of the system as a formula ϕ in that logic, and using an automated procedure to check whether ϕ is true in M. However, while there has been considerable work on the formal verification of software systems and on logics of agency, it has proved difficult to bring this work to bear on verification of agent architectures and programs. On the one hand, it can be difficult to specify and verify relevant properties of agent programs using conventional formal verification techniques, and on the other, standard epistemic logics of agency (e.g., [16]) fail to take into account the computational limitations of agent implementations.

Since an agent program is a special kind of program, logics intended for the specification of conventional programs can be used for specifying agent programming languages. In this approach we have some set of propo-

sitional variables to encode the agent's state, and, for example, dynamic or temporal operators for describing how the state changes as the computation evolves. However, for agents based on the Belief-Desire-Intention model of agency, such an approach fails to capture important structure in the agent's state which can be usefully exploited in verification. For example, we could encode the fact that the agent has the belief that p as the proposition u_1, and the fact that the agent has the goal that p as the proposition u_2. However such an encoding obscures the key logical relationship between the two facts, making it difficult to express general properties such as 'an agent cannot have as a goal a proposition which it currently believes'. It therefore seems natural for a logical language intended for reasoning about agent programs to include primitives for the beliefs and goals of the agent, e.g., where Bp means that the agent believes that p, and Gp means that the agent has a goal that p.

Given that a logical language intended for reasoning about agent programs should include primitives for the beliefs and goals of an agent, what should the semantics of these operators be? For example, should the belief operator satisfy the KD45 properties? In our view, it is critical that the properties of the agent's beliefs and goals should be grounded in the computation of the agent (in the sense of [31], that is, there should be a clear relationship between the semantics of beliefs and goals and the concrete computational model of the agent). If the agent implements a full classical reasoner (perhaps in a restricted logic), then we can formalise its beliefs as closed under classical inference. However if the agent's implementation simply matches belief literals against a database of believed propositions without any additional logical reasoning, we should not model its beliefs as closed under classical consequence. The notion of 'computationally grounded' logics is discussed in more detail in the next section.

3 Standard epistemic logic is not computationally grounded

Since the first BDI logics such as [13] and [27], the knowledge and beliefs of AI agents have been modelled using epistemic modal logics with possible worlds semantics. An agent i believes a formula ϕ in a possible world or state s if ϕ is true in all states s' which are belief-accessible from s. For knowledge, the accessibility relation is usually assumed to be an equivalence relation between states, intuitively meaning that the agent cannot tell whether the actual state is s or one of the other knowledge-accessible states. This relation is often referred to as the 'indistinguishability relation' of agent i and denoted by $\sim_i$. A more concrete version of the possible worlds semantics are interpreted systems introduced in [16], where each state is an n-tuple of the agents' local states and the state of the environment (assuming the system consists of n agents and an environment) and $s \sim_i s'$ holds if the local state of agent i is the same in s and s'. The logic over interpreted systems has temporal operators in addition to the epistemic ones, and the formulas are interpreted over computational runs (sequences of states). The epistemic logics based on this semantics have attractive formal properties. However, they suffer from two main problems: the problem of correctly ascribing beliefs to

an agent, and the problem of logical omniscience.

The problem of belief ascription is concerned with the difficulty of determining what an agent's beliefs are at a given point in its execution. Many agent designs do not make use of an explicit representation of beliefs within the agent. For example, the behaviour of an agent may be controlled by a collection of decision rules or reactive behaviours which simply respond to the agent's current environment. However when modelling the agent, it can still be useful to view the agent as having beliefs. For example, when modelling an agent with a reactive architecture which does not explicitly represent beliefs, we may say that "the agent believes there is an obstacle to the left" and "if the agent believes there is an obstacle to the left, it will turn to the right". However, for this to be possible, we need some principled way of deciding what the agent believes.

However, even when agents do represent beliefs explicitly, the mapping between the agent's belief state and the logical model of the agent is not straightforward. As noted by van der Hoek and Wooldridge [32, p.149] "possible worlds semantics are generally *ungrounded*. That is, there is usually no precise relationship between the abstract accessibility relations that are used to characterise an agent's state, and any concrete computational model." This makes it difficult to use BDI logics for specifying agent systems or to use model-checking tools and algorithms to model-check a particular agent program, since one would need to somehow extract from the program the belief accessibility relations for generating a logical model for use in model-checking. "Because, as we noted earlier, there is no clear relationship between the BDI logic and the concrete computational models used to implement agents, it is not clear how such a model could be derived."[32, p. 153] This problem does surface in model-checking BDI agent programs; see, for example, [8], where the beliefs of an agent which is intended to implement the LORA architecture [34] (which uses standard semantics for beliefs) are modelled syntactically as a finite list of formulas rather than using an accessibility relation. Similar concerns about a gap between BDI logics and concrete agent programs, or the lack of groundedness, were raised by Meyer in [23].

One consequence of this lack of computational grounding is that epistemic logics based on possible worlds semantics model agents as logically omniscient reasoners: they believe/know all tautologies and they believe/know all logical consequences of their beliefs/knowledge ($B_i\top$ and $B_i\phi \wedge B_i(\phi \rightarrow \psi) \rightarrow B_i\psi$ are tautologies of any logic with a modal operator B_i defined as truth in all i-accessible worlds). In effect, agents are modelled as perfect logical reasoners with unlimited computational powers. This is problematic when attempting to build realistic models of agent behaviour, where the time required by the agent to solve a problem is often of critical importance.

Some authors (for example, [26], [25]) argue that unlike the possible worlds structures, interpreted systems can be seen as a grounded semantics for intensional logics. The following arguments are paraphrased from [25, p.36]

- since the semantics of interpreted systems refers to computational runs, a system description in terms of runs (using local states, protocols, etc.) immediately provides a logical model to evaluate formulae;

- epistemic properties are based on the equivalence of local states (which is a concrete computational notion); and
- local states could be represented as e.g. arrays of variables, thereby allowing for a 'fine grained' description of agents.

The last point and the examples of modelling multi-agent systems such as Dining Cryptographers given in [26, 25] suggest the following way of ascribing knowledge to agents. The local state of the agent (values of variables) determines what the agent 'knows'. For example, a propositional variable $paid_i$ may mean that a variable v_1 in agent i's state has value $Paid$ (see [26]). Clearly, we want that when the agent i is in the same local state s_i^0 where $v_1 = Paid$, $K_i paid_i$ holds. And this works out with the interpreted systems definition of knowledge: since $paid_i$ holds in all global states where agent $i's$ state is s_i^0, in all those global states $K_i paid_i$ holds.

However, while at first sight this semantics may appear to be computationally grounded, we argue that it is a very roundabout way of defining an agent's knowledge. The indistinguishability relation $\sim_i$ between global states is used for the truth definition of K_i formulas, so to determine the truth of such formulas we need to examine all global states related by $\sim_i$, which adds significant complexity to, for example, the model-checking problem. This additional (and unnecessary) complexity is purely an artefact of the possible worlds truth definition of K_i. From the start we decided that what the agent actually knows depends on the properties of the agent's local state and we should really only have to examine the agent's state.

Another, more serious, artefact of this truth definition, is that $K_i\phi$ holds not only for the formulas ϕ which do correspond to some properties of the agent's state (the real or explicit knowledge of the agent), but also for a host of other formulas. Among those additional formulas are tautologies and logical consequences of the real knowledge of the agent such as $K_i(\neg(paid_i \wedge \neg paid_i))$, consequences by introspection (such as $K_i K_i K_i K_i paid_i$ and $K_i K_i K_i K_i \neg K_i \neg paid_i$) and, even more paradoxically, formulas talking about the global properties of the system. For example, if there is just one global state s^0 where agent i's local state is s_i^0, and s^0 has a single successor s^1, then i 'knows' precisely what the next global state looks like. For example, suppose some proposition q is true in s^1. Then $\bigcirc q$ ('in the next state, q') is true in s^0. Since s^0 is the only state $\sim_i$-accessible from s^0, agent i knows that in the next state q: $K_i \bigcirc q$, and similarly for all the other formulas true in s^1. This is by no means a grounded knowledge ascription. Even if the system is entirely deterministic, agent i does not necessarily have any knowledge of this.

4 Other approaches

Problems such as logical omniscience which arise when interpreted systems are used to model resource-bounded reasoners have been known for a long time, and some proposed solutions are described in [16]. One of the solutions is termed syntactic in [16]: instead of using a possible worlds truth definition for the knowledge modality K_i, in each state we get essentially a syntactic assignment of formulas agent i believes in that state. For consistency with

what follows we will denote this set of formulas as $\mathcal{A}_i(s)$. The truth definition for $K_i\phi$ in state s of a model M will then become $M, s \models K_i\phi$ iff $\phi \in \mathcal{A}_i(s)$. Clearly, this truth definition makes K_i entirely free of the problem of logical omniscience; however in [16] it is argued that its shortcoming is the lack of any interesting properties of K_i. They prefer a related approach, of combining K_i defined using the possible worlds definition with a new syntactic operator A_i standing for 'awareness'. K_i is the standard (true in all $\sim_i$) notion of knowledge (called 'implicit knowledge' in [16]); $A_i\phi$ is true if $\phi \in \mathcal{A}_i(s)$ and this only means that i is aware of ϕ, not that i necessarily knows that ϕ; and $X_i\phi =_{df} K_i\phi \wedge A_i\phi$ means that i explicitly knows ϕ. Clearly, explicit knowledge is also not closed under consequence since i may not be aware of some of the consequences.

Another related approach to 'fixing' the idealisation inherent in the possible worlds definition of knowledge is the concept of algorithmic knowledge [16]. Instead of some arbitrary syntactic set $\mathcal{A}_i(s)$, we assume that each state s comes equipped with an algorithm $alg_i(s)$ and agent data (used by the algorithm) $data_i(s)$. An agent algorithmically knows a formula ϕ in s if the output of $alg_i(s)$ for ϕ is 'yes'. In [16], a number of interesting questions are raised, for example how to relate the computation of the knowledge answering algorithm to the rest of the system dynamics, but the authors decided to keep the two issues separate: the algorithm computation is assumed to be instantaneous and not included in the rest of the system transitions. Subsequent work on algorithmic knowledge, see for example [24], continues to adopt this 'closure under the algorithm' condition for the algorithmic knowledge, although beliefs are represented as tokens, and the algorithm as a set of rewriting rules.

5 Syntactic belief ascription

In this section we present *syntactic belief ascription* as an alternative approach to computationally grounded belief ascription. We distinguish between beliefs and reasoning abilities which we ascribe to the agent ('the agent's logic') and the logic we use to reason *about* the agent. In this we follow, e.g., [21, 20, 17]. Our approach grounds the ascription of belief in the state of the agent and allows us to explicitly model the computational delay involved in updating the agent's state. Our logic for reasoning about the agent's beliefs is grounded in a concrete computational model in the sense of [33]. However, unlike [33, 29] we choose not to interpret the agent's beliefs as propositions corresponding to sets of possible states or runs of the agent's program, but syntactically, as formulas 'translating' some particular configuration of variables in the agent's internal state. One of the consequences of this choice is that we avoid modelling the agent as logically omniscient. This has some similarities with the bounded-resources approach of [15] and more recent work such as [14, 4, 1]. We first consider grounded belief ascription (given an agent state, how to ascibe beliefs to it in a grounded way), and then discuss closure assumptions (what assumptions is it safe to make concerning the closure of the agent's belief set under the agent's inferential capabilities).

5.1 Grounded belief ascription

Similarly to interpreted systems, we consider states to be $n + 1$-tuples of local states of n agents and the state of the environment, in other words, $s = (s_1, \ldots, s_n, e)$. We describe the properties of the system in a language built from a set of propositional variables $\mathcal{P}$. Beliefs ascribable to an agent i are a finite set of literals (variables or their negations) over $\mathcal{P}$ which we will denote L_i. Following, for example, Rosenschein and Kaelbling [28], we assume that each agent's state consists of finitely many 'memory locations' $\mathbf{l_1}$,...,$\mathbf{l_m}$, and that each location $\mathbf{l_j}$ can contain (exactly) one of finitely many values, $v_{j_1}, \ldots, v_{j_k}$. For example, we could have a location $\mathbf{l_t}$ for the output of a temperature sensor which may take an integer value between -50 and 50. Based on those values, we can ascribe beliefs about the external world to the agent: for example, based on $\mathbf{l_t}$ = 20 we ascribe to the agent a belief that the outside temperature is 20 C. Each literal in L_i corresponds to the fact that a given memory location $\mathbf{l_j}$ (or set of memory locations) has a given set of values, but 'translates' this into a statement about the world. We assume a mapping $\mathcal{A}_i$ assigning to each state s a set of propositional variables and their negations which form beliefs of agent i in state s. Note that this 'translation' is fixed and does not depend on the truth or falsity of the formulas in the real world. In general, there is no requirement that $\mathcal{A}_i$ be consistent; if a propositional variable and its negation are associated with two different memory locations (e.g., in an agent which has two temperature sensors) then the agent may simultaneously believe that p and $\neg p$[1]. $\mathcal{A}_i$ does not have to map a single value to a single belief, for example, all values of $\mathbf{l_t} > 20$ could be mapped to a single belief that it's "warm". Conversely, we do not assume that for every propositional variable $p \in \mathcal{P}$, either p or $\neg p$ belong to $\mathcal{A}_i$; if a location $\mathbf{l_j}$ has no value (e.g., if a sensor fails) or has a value that does not correspond to any proposition, then the agent may have no beliefs about the outside world at all. Other intentional notions such as goals can be modelled analogously to beliefs, i.e., by introducing an explicit translation from the contents of the agent's state into the set of goals. We elaborate belief and goal ascription using the notion of a memory location rather than assuming that agents have an internal representation of beliefs or goals e.g., as a list of literals, for reasons of generality. The ascription mechanism described above is applicable to arbitrary agents, not only those with an explicit internal representation of beliefs and goals. In certain sense, we can say that this ascription $\mathcal{A}_i(s)$ corresponds to the agent's 'awareness' of the facts explicitly represented in agent i's local state in the global state s.

Our aim is to model the transitions of the agent-environment system as a kind of Kripke structure and express properties of the agents in a modal logic. We consider transition systems similar to the interpreted systems of [16], except that the beliefs of agents are modelled as a local property of each agent's state using syntactic assignment $\mathcal{A}_i$ corresponding to agent i's beliefs. The state of the environment e corresponds to a classical possible world, or a complete truth assignment to propositional variables in $\mathcal{P}$. Agent

[1] This assumes that the agent's program is using beliefs about the outside world, rather than indirect beliefs about sensor reading; in the latter case of course there would be no contradictory beliefs, since different sensors would have different propositions associated with them.

i believes that p in state s, $M, s \models B_i p$, if $p \in \mathcal{A}_i(s)$. Our proposal is technically equivalent to the syntactic model of belief in [16]. The difference between our approach and [16], is that while a syntactic assignment in [16] is an arbitrary set of formulas, we show how to ground this set in the set of values of variables in the agent's state. Note that this truth definition for B_i does not give rise to any interesting logical properties of B_i, e.g., to KD45 axioms. This is intentional: we do not want our agents to be logically omniscient and the logical properties of agent's beliefs should be determined by the agent's architecture and program. However, the agent's state changes as the agent executes its program; it could be argued that we may assume that some computation of 'consequences' of the agent's beliefs takes so little time that we can safely assume that the set of beliefs is closed with respect to the agent's 'internal logic' (this argument is made in favour of the closure under algorithmic knowledge in [16]). This is the topic of the next section.

5.2 Deductive closure assumptions

Clearly, any assumptions concerning deductive closure of the agent's beliefs should be based on the agent's program and on the requirements of modelling. One could argue that if the agent only ever 'tests' its beliefs to check whether it believes p or $\neg p$, and belief ascription with respect to p is correct, it is safe to ascribe to the agent a set of beliefs closed with respect to full classical logic since the agent program does not make any choices depending on the presence of all the extraneous beliefs (logical tautologies and the like). However, we would argue that beyond such relatively trivial agent programs, the deductive closure assumptions should be taken seriously since they may result in incorrect belief ascription.

We argue that for any agent which answers queries or choses actions depending on its beliefs, the assumption of deductive closure of beliefs is only safe if

1. the closure is with respect to the agent's real 'internal logic' or query answering algorithm implemented by the agent program (so the postulated consequences are actually derivable)
2. the requirements of modelling allow for a reasonably coarse granularity of time, and it is reasonable to assume that the agent's deductive algorithm completes within a finite and reasonably short period of time.

For example, it may make sense to model beliefs of a forward-chaining rule-based agent as closed under applying the forward chaining procedure to a finite set of ground beliefs, provided that it does not take a long time to terminate and we are not concerned with the precise timing of the agent's response to a query. Under such conditions, it is reasonable to model the agent's beliefs using the deductive algorithmic knowledge approach [24]. Note that although the set of beliefs of such an agent is deductively closed, it is deductively closed with respect to a very weak logic (basically, a logic containing universal quantifier elimination and modus ponens, which is much weaker that the full classical logic).

Consider again a forward-chaining rule-based agent but assume that its deductive procedure is not guaranteed to terminate (for example, the agent's

rules contain arithmetic expressions). In this case, even if modelling allows for quite coarse granularity of time, it would not be safe to model beliefs of the agent as deductively closed (since some of the logical consequences will never be derived in reality but will be ascribed to the agent by the deductive closure assumption). In this case, the deductively closed set of beliefs, even in the agent's own 'logic' (its 'implicit knowledge') will be an idealisation, and the algorithmic knowledge approach could be made to work (for a given granularity of modelling) by amending the agent's forward-chaining algorithm with a 'timeout' corresponding to the granularity of modelling: e.g. if the agent's state is 'sampled' at 10 minute intervals then the agent's beliefs can be modelled as closed with respect to applying the forward-chaining algorithm for 10 minutes.

Finally, arguably the safest way of ascribing beliefs to a resource-bounded agent is not to make any closure assumptions for the set of agent's beliefs, and to model each inference step in the agent's internal logic as an explicit transition of the system; this choice gives rise to dynamic syntactic logics such as [14, 1, 4].

6 Verifying agent programs

In this section we briefly outline some applications of syntactic epistemic logics in verifying agent programming languages.

6.1 Theorem proving

SimpleAPL is simplified version of the BDI-based agent programming languages 3APL and 2APL, see, e.g., [9]. SimpleAPL programs have explicit data structures for beliefs and goals, and a program is specified in terms of the agent's beliefs, goals and planning goal rules which specify which plans the agent should adopt given its goals and beliefs. An approach to verification of SimpleAPL programs based on syntactic epistemic logic was described in [2] and extended to verification of agent programs under different deliberation strategies in [3]. Given the explicit representation of beliefs, belief ascription for SimpleAPL agents is straightforward: $B\,p$ (the agent believes that p) is true if p is present in the belief base of the agent. SimpleAPL agents do not do any inference, so the set of beliefs is not closed under any inference rules. Verification in [2] is done by theorem proving; an agent program is axiomatised in Propositional Dynamic Logic (PDL) extended with syntactic belief and goal operators, and a statement such as 'all executions of this program starting in a state with initial beliefs $p_1, \ldots, p_n$ and goals $\kappa_1, \ldots, \kappa_m$ will achieve the agent's goals' can be verified by checking whether the following formula is derivable from the axiomatisation of the agent program:

$$\bigwedge_{i=n}^{i=1} Bp_i \wedge \bigwedge_{j=m}^{j=1} G\kappa_j \rightarrow [prog] \bigwedge_{j=m}^{j=1} B\kappa_j$$

where $prog$ is a translation of the agent's program into PDL with syntactic belief and goal operators.

As an example, consider the following example of a vacuum cleaner agent. Actions in SimpleAPL are specified using pre and postcondition pairs (intuitively, what the agent should believe before it can execute an action, and what its beliefs are expected to be after it executes an action). Suppose the agent has the following actions:

```
{room1} moveR {-room1, room2}
{room2} moveL {-room2, room1}
{room1, battery} suck {clean1, -battery}
{room2, battery} suck {clean2, -battery}
{-battery} charge {battery}
```

We will abbreviate goals and beliefs of the agent as: c_i for $\texttt{clean}_\texttt{i}$, r_i for $\texttt{room}_\texttt{i}$, b for `battery`, and actions s for `suck`, c for `charge`, r for `moveR`, l for `moveL`. Suppose the program of the agent contains the following planning goal rules:

```
c1 <- b | if r1 then s else l; s
c2 <- b | if r2 then s else r; s
   <- -b | if r2 then c else r; c
```

These rules allow the agent to select an appropriate plan to achieve a goal given its current beliefs. For example, the first rule can be read as "if the goal is to clean room 1, and the battery is charged, adopt the following plan: if in room one, suck, else move left and then suck".

The corresponding PDL program expression $prog$ is:

$$\begin{aligned}\texttt{prog} \quad =_{df} \quad & ((Gc_1 \wedge Bb)?; (Br_1?; s) \cup (\neg Br_1?; l; s)) \cup \\ & ((Gc_2 \wedge Bb)?; (Br_2?; s) \cup (\neg Br_2?; r; s)) \cup \\ & (\neg Bb?; (Br_2?; c) \cup (\neg Br_2?; r; c))\end{aligned}$$

Some example axioms:

$Bp \rightarrow \neg Gp$ (for every variable p: the agent does not have as a goal something that it believes has been achieved)

$Br_2 \wedge Bb \wedge Gc_2 \rightarrow [s](Bc_2 \wedge \neg Bb \wedge Br_2)$ (corresponds to the pre and postconditions of the `suck` action).

We used MSPASS and pdl_tableaux theorem provers to prove the following properties:

- if the agent has goals to clean rooms 1 and 2, and starts in the state where its battery is charged and it is in room 1, it can reach a state where both rooms are clean: $Gc_1 \wedge Gc_2 \wedge Bb \wedge Br_1 \rightarrow \langle prog^3 \rangle (Bc_1 \wedge Bc_2)$ (where $prog^3$ stands for $prog$ repeated three times)
- the agent is guaranteed to achieve its goal (after 3 iterations of the program) $Gc_1 \wedge Gc_2 \wedge Bb \wedge Br_1 \rightarrow [prog^3](Bc_1 \wedge Bc_2)$

The logic sketched above is grounded in the agent programming language because its models correspond to the agent's operational semantics. It can be used to specify and automatically verify properties of SimpleAPL programs.

6.2 Model-checking

In the previous section, we sketched an approach to verifying agent programs using theorem proving. Another approach is to use a model-checker. There are two main strands of work in model-checking multi-agent systems and agent programs which are exemplified by: model-checking based on 'standard' BDI logics e.g., [22] and model-checking based on a syntactic interpretation of beliefs e.g., [11, 10, 30].

The only model-checker which 'understands' epistemic operators (knowledge) is MCMAS [22]. MCMAS allows checking of properties along both temporal and knowledge accessibility relations. Unfortunately, it is nontrivial to relate 'possible worlds' knowledge to the knowledge or beliefs of implemented BDI agents.

An alternative approach is to model-check an agent programming language treating belief as a syntactic modality. This is the approach implicitly taken in [11] for the verification of AgentSpeak(F) programs. Belief, desire and intention are defined in terms of the operational semantics of AgentSpeak(F):

- an agent believes ϕ if ϕ is present in its belief base
- an agent intends ϕ if ϕ is an achievement goal that appears in the agent's set of intentions – i.e., in the agent's currently executing or suspended plans.

AgentSpeak(F) programs are translated into the Promela modelling language of the Spin model checker. Properties to be model-checked are expressed in a simplified BDI logic translated into the LTL-based property specification language used by Spin. BDI modalities are mapped onto the AgentSpeak(F) structures implemented as a Promela model. So, even though Bordini et al. [11] do not mention the problems with the standard semantics of belief, or dwell on using the syntactic approach to beliefs rather than the LORA framework based on the standard epistemic semantics which they officially adopt, the fact is that they do use a syntactic approach. We argue that this is inevitable in verification of a real agent programs. A similar approach (using syntactic or 'shallow' modalities) is adopted in [10] and [30].

The work on model-checking agent programs using syntactic approaches mentioned in this section does not model agent's deriving consequences from its beliefs explicitly. However, in other work where the main concern is with the time required for the agents to produce a response to a query, we did use model-checking over systems where transitions correspond to agents applying inference rules (usually, forward-chaining rule firing): see [4, 7] for more details.

7 Conclusion

'Standard' BDI logics allow properties of beliefs and other intentional attitudes of AI agents to be formalised. The resulting specifications can be model checked using model checkers such as MCMAS. However it is not clear how to implement agents based on these specifications; in particular, it is not clear what corresponds to belief and goal accessibility relations in the

agent programming language or the implemented agent. On the other hand, 'syntactic' BDI logics allow more accurate modelling of AI agents. We can verify properties of real agent programs at the belief and goal level (as opposed to simply verifying the agent program as just a computer program).

Acknowledgements Natasha Alechina and Brian Logan were supported by the Engineering and Physical Sciences Research Council [grant number EP/E031226].

References

[1] Thomas Ågotnes and Natasha Alechina. The dynamics of syntactic knowledge. *Journal of Logic and Computation*, 17(1):83–116, 2007.

[2] Natasha Alechina, Mehdi Dastani, Brian Logan, and John-Jules Ch. Meyer. A logic of agent programs. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI 2007)*, pages 795–800. AAAI Press, 2007.

[3] Natasha Alechina, Mehdi Dastani, Brian Logan, and John-Jules Ch. Meyer. Reasoning about agent deliberation. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 16–26, Sydney, Australia, September 2008. AAAI.

[4] Natasha Alechina, Mark Jago, and Brian Logan. Modal logics for communicating rule-based agents. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 322–326. IOS Press, 2006.

[5] Natasha Alechina and Brian Logan. Ascribing beliefs to resource bounded agents. In Cristiano Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, pages 881–888, Bologna, Italy, July 2002. ACM Press.

[6] Natasha Alechina and Brian Logan. A logic of situated resource-bounded agents. *Journal of Logic, Language and Information*, 18(1):79–95, 2009.

[7] Natasha Alechina, Brian Logan, Nguyen Hoang Nga, and Abdur Rakib. Verifying time, memory and communication bounds in systems of reasoning agents. *Synthese*, 169(2):385–403, July 2009.

[8] Rafael Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. State-space reduction techniques in agent verification. In Nicholas R. Jennings, Carles Sierra, Liz Sonenberg, and Milind Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004)*, pages 896–903, New York, NY, 2004. ACM Press.

[9] Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer, 2005.

[10] Rafael H. Bordini, Louise A. Dennis, Berndt Farwer, and Michael Fisher. Automated verification of multi-agent programs. In *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), 15-19 September 2008, L'Aquila, Italy*, pages 69–78. IEEE, 2008.

[11] Rafael H. Bordini, Michael Fisher, Carmen Pardavila, and Michael Wooldridge. Model checking AgentSpeak. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 409–416, New York, NY, USA, 2003. ACM.

[12] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[13] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[14] Ho Ngoc Duc. Reasoning about rational, but not logically omniscient, agents. *Journal of Logic and Computation*, 7(5):633–648, 1997.

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

[16] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.

[17] Ronald Fagin, Joseph Y. Halpern, and Moshe Y. Vardi. A non-standard approach to the logical omniscience problem. *Artificial Intelligence*, 79(2):203–240, 1996.

[18] Michael P. Georgeff, Barney Pell, Martha E. Pollack, Milind Tambe, and Michael Wooldridge. The belief-desire-intention model of agency. In Jörg P. Müller, Munindar P. Singh, and Anand S. Rao, editors, *Intelligent Agents V, Agent Theories, Architectures, and Languages, 5th International Workshop, (ATAL'98), Paris, France, July 4-7, 1998, Proceedings*, volume 1555 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1999.

[19] J. Hintikka. *Knowledge and belief*. Cornell University Press, Ithaca, NY, 1962.

[20] K. Konolige. *A Deduction Model of Belief*. Morgan Kaufmann, San Francisco, Calif., 1986.

[21] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence, AAAI-84*, pages 198–202. AAAI, 1984.

[22] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, 21st*

International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer, 2009.

[23] John-Jules Ch. Meyer. Our quest for the holy grail of agent verification. In Nicola Olivetti, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, 16th International Conference, TABLEAUX 2007, Aix en Provence, France, July 3-6, 2007, Proceedings*, volume 4548 of *Lecture Notes in Computer Science*, pages 2–9. Springer, 2007.

[24] Riccardo Pucella. Deductive algorithmic knowledge. *Journal of Logic and Computation*, 16(2):287–309, 2006.

[25] Franco Raimondi. *Model-checking multi-agent systems*. PhD thesis, Department of Computer Science, University College London, University of London, 2006.

[26] Franco Raimondi and Alessio Lomuscio. A tool for specification and verification of epistemic properties in interpreted systems. *Electronic Notes in Theoretical Computer Science*, 85:176–191, 2004.

[27] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

[28] S. J. Rosenschein and L. P. Kaelbling. A situated view of representation and control. *Artificial Intelligence*, 73:149–173, 1995.

[29] N. Seel. The 'logical omniscience' of reactive systems. In *Proceedings of the Eighth Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB'91)*, pages 62–71, Leeds, England, 1991.

[30] Doan Thu Trang, Brian Logan, and Natasha Alechina. Verifying Dribble agents. In Matteo Baldoni, Jamal Bentahar, M. Birna van Riemsdijk, and John Lloyd, editors, *Declarative Agent Languages and Technologies VII, 7th International Workshop, DALT 2009, Budapest, Hungary, May 11, 2009. Revised Selected and Invited Papers*, volume 5948 of *Lecture Notes in Computer Science*, pages 244–261, 2010.

[31] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):133–157, 2003.

[32] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):135–159, 2003.

[33] Michael Wooldridge. Computationally grounded theories of agency. In E. Durfee, editor, *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, pages 13–20. IEEE Press, 2000.

[34] Michael Wooldridge. *Reasoning About Rational Agents*. MIT Press, 2000.