Verification of Agent-Oriented Situated Systems: A Model-Theoretic Approach

Anand S. Rao and *Michael P. Georgeff*
Australian Artificial Intelligence Institute
1 Grattan Street, Carlton
Victoria 3053, Australia

Phone: (+61 3) 663-7922, Fax: (+61 3) 663-7937
Email: anand@aaii.oz.au and georgeff@aaii.oz.au

Abstract

The study of situated systems that are capable of reactive and goal-directed behaviour has received increased attention in recent years. One important class are the so-called agent-oriented systems, which model the system as a rational agent with certain mental attitudes that determine its decision-making and acting capabilities. This approach has led to the development of expressive, but computationally intractable, logics for describing or specifying the behaviours of situated systems. In this paper, we present three propositional variants of such logics, with different expressive power, and analyze the computational complexity of verifying if a given property is satisfied by a given abstract situated program. The linear time and polynomial time complexity of the verification algorithms for two of these logics provides encouraging results with respect to the practical use of such logics for verifying situated systems.

1 Introduction

The study of systems that are situated or embedded in a changing environment has been receiving considerable interest within the knowledge representation and planning communities. The primary characteristic of these systems is their dynamic and resource-bounded nature. In particular, situated systems need to provide an appropriate balance between time spent deliberating and time spent acting. If the time spent on deliberation is too long, the ability of the system to complete its tasks may be seriously affected. On the other hand, acting with very little deliberation may lead to a system that is short-sighted and overly reactive.

A number of different approaches have emerged as candidates for the study of such situated systems [2, 3, 5, 18, 21, 22, 23]. Some of the most interesting of these are agent-oriented architectures, in which the system is viewed as a rational agent having certain *mental attitudes* that influence its decision making and determine its behavior. The simplest of these architectures, called a BDI architecture, is based on attitudes of *belief*, *desire* and *intention*. The first two attitudes represent, respectively, the information and evaluative states of the agent. The last represents decisions the agent has made at the current or previous time, and is critical for achieving adequate or optimal performance when deliberation is subject to resource bounds [1, 13].

A primary aim of the study of rational agency, at least within Artificial Intelligence, is to build artificial agents that can exhibit rational behaviour and can interact rationally with other artificial agents and human beings in a number of different environments. While it may be ultimately possible to build a single monolithic agent that can perform all tasks in all environments, a more achievable goal is to build specific agents that can excel in specific environments by exhibiting the most appropriate behaviour for that type of environment. We thus characterize the study of rational agency as the study of environments (E), the study of the internal mental state of agents (S), and the rational behaviours (B) that agents situated in these environments can generate. The agent, in these terms, is modeled by a function, $A: S \times E \rightarrow B$.

There are, in fact, three elements to this study. First, *specification* involves the description of a particular environment $e \in E$, an internal mental state $s \in S$, and a set of rational behaviours $b \subset B$. Second, *design* involves the construction of an agent $a \in A$ with internal mental state s which, when situated in an environment e', will generate the set of behaviours b'. The input for the design process is the specification triple (s, e, b). Third, *verification* is the process by which the design of agent a is verified to meet the specification of the agent. If the environment e is the same as e' then the verification process requires that the observed behaviours b' be equal to the specified behaviours b.

Digital circuits, situated automata, and logic are some of the alternatives for specifying, designing, and verifying artificial agents. Our primary interest is in the use of logic as a tool for studying all three aspects of rational agency.

For the purpose of specification, a modal temporal logic can be used. To be adequate for interesting environments and the agents that are designed to populate

them, it is necessary to use a first-order temporal logic, either point- or interval-based. In addition, to specify the internal mental states of the agent one requires *attitude modalities* for beliefs, desires, and intentions, and means for describing complex plans and actions.

In addition to an expressive language, the specification should have a well-defined semantics. Of particular interest are the interactions between different attitude modalities. These interactions can be characterized as *static* and *dynamic* interactions. Static interactions describe the relationships among the different mental attitudes at the same time point, whereas dynamic interactions describe how these mental attitudes evolve over time. Such interactions are not very well understood and only some aspects of these interactions, such as how beliefs evolve over time (belief revision), have been formalized.

The semantics of the specification and the various interactions can be captured both model-theoretically and proof-theoretically. The model-theoretic semantics introduces various primitive semantic entities and describes the meaning of other complex entities by reducing them to these primitive ones. The interactions among the mental attitudes can be represented as constraints on the semantic relations that model these attitudes. In the proof-theoretic semantics, non-temporal axioms that describe the static interactions and temporal axioms that describe the dynamic interactions of mental attitudes are given.

The design of an agent from the specification can be carried out in at least three ways. In the *model-theoretic design* of an agent, an initial partial model of the mental state of an agent is generated from the specification. The various processes of an agent that manipulate the mental attitudes are implemented as algorithms that modify and extend this initial partial model. Thus, as the mental attitudes of an agent change and evolve over time the initial partial model also evolves over time. The major disadvantage with this approach is managing the size of the partial model.

The *proof-theoretic design* of an agent involves building a theorem prover for the specification logic. The various processes of an agent are represented as axioms. The major problem with this approach is the computational complexity of proving theorems in the expressive specification logic.

Finally, a *program-based design* considers the mental attitudes of an agent as data structures and implements the various processes that manipulate these mental attitudes as programming code. The static and dynamic interactions are captured as functions that manipulate these data structures (mental attitudes). This approach is probably the easiest and the most widely adopted of the three approaches.

Verification of agent design with the specification is one of the most important and yet least researched topic in the study of rational agency. A model-theoretic design can be verified by checking if the final complete model (the one obtained at the end of observing all the interesting behaviours one wants to observe) satisfies the behavioural properties as given by the specification. Depending on the expressive power of the specification logic, model-checking can be performed in linear to polynomial time.

A proof-theoretic design can be verified by proving the behavioural properties of the specification as theorems of the proof theory with the initial mental state as the set of assumptions. Verification using this approach is like to have an exponential or higher degree of complexity.

Of all the three designs, the program-based design is the most difficult to verify. One has to establish correspondences between the data structures at the beginning of each interpreter cycle with the static interactions of the mental attitudes, and also the change of data structures over a number of interpreter cycles with the dynamic interactions of the mental attitudes.

A majority of the work in the study of such agent-oriented systems have concentrated on the specification or characterization of rational agents and their behaviours under different environmental conditions. Most of these logics use linear or branching temporal structures, are often first-order, and tend to have a rich repertoire of modal operators to model beliefs, goals, intentions, commitment, ability, actions, and plans.

However, the design of situated systems has so far had little connection with these formalisms. Although some systems have been designed and built based upon the philosophy of rational agents [10], the linkage between the formal specification and the design is weak. Similarly, little has been done in verification of situated systems. As more and more AI-based situated systems are being tested and installed in safety-critical applications, such as air-traffic management, real-time network management, and power-system management, the need to verify and validate such systems is becoming increasingly important.

A detailed exposition of all three aspects of the study of rational agents is beyond the scope of this paper. We here briefly describes three logics, with different expressive power, for specifying rational agents and their interactions with a dynamic environment. The complexity of verifying such agent-oriented situated systems is then discussed. Issues related to the modeling of different types of rational agents and the practical design of agent-oriented systems are dealt with elsewhere [18, 19].

The outline of the paper is as follows. Section 2 describes the semantic model. Section 3 presents two branching time BDI logics and models commitment. The problem of verification in both these logics is described in Section 4. Section 5 defines a third language with intermediate expressive power and provides polynomial time algorithms for model checking. Using an example, Section 6 shows how one can verify temporal and commitment properties of situated systems in polynomial time. Finally, we conclude in Section 7 by comparing our work with related effort and highlighting the contributions of this paper.

2 Overview

Situated agents can be viewed as concurrent systems of processes. The concurrent execution of a family of individual processes can be modeled by the nondeterministic interleaving of the atomic actions of each process. In such a model, the actions taken by one process and the actions of another process are not structurally distinguishable. The nondeterminism resulting from the manner in which the processes are interleaved is represented as a time point with multiple successors in a branching-time tree structure. Each, possibly infinite, execution sequence of a concurrent program is represented as a computation path of the tree structure.

A rational agent situated in an uncertain and dynamic world needs to distinguish the uncertainty or *chance* inherent in the environment as well as the *choice* of actions available to it at each and every time point. In the study of rational agents, unlike concurrency theory, one is particularly interested in analyzing how an agent can bring about the future that it desires. As the agent does not have direct control over the environment, but has direct control over the actions it can perform, it is desirable to separate the agent's choice of action (over which it has control) from its view of the environment (over which it has no control). Also, unlike concurrency theory, there is no single view of the environment; each agent can have its own view of the environment and of other agents' mental states which may not coincide with the actual environment nor the actual mental states of these agents. These different views of the world can be more effectively modeled within a possible-worlds framework than as a monolithic branching tree structure

Hence, we adopt a *possible worlds branching-time tree structure* in which there are multiple possible worlds and each possible world is a branching-time tree structure. Multiple possible worlds model the chance inherent in the environment as viewed by the agent and are a result of the agent's lack of knowledge about the environment. But within each of these possible worlds, the branching future represents the choice of actions available to the agent.

A particular time point in a particular world is called a *situation*. For each situation we associate a set of *belief-accessible*, *desire-accessible*, and *intention-accessible* worlds; intuitively, those worlds that the agent *believes* to be possible, *desires* to bring about, and *commits* to achieving, respectively. We require that an agent's intentions be consistent with its adopted desires, and that its desires be believed to be achievable [16]. This is captured semantically by requiring that for each belief-accessible world there exists a sub-world that is desire-accessible and, in turn, for each desire-accessible world there exists a sub-world that is intention-accessible [18].

One of the important properties in reasoning about concurrent programs and the processes they describe is the notion of *fairness*. Fairness or fair scheduling assumptions specify when an individual process in a family of concurrent processes must be scheduled to execute next. A number of different fairness assumptions have been analyzed in the literature [7, 9, 14]. A commonly used fairness assumption is that a process must be executed infinitely often. A system of concurrent processes can thus be viewed as a branching-time tree structure, with fairness and starting conditions. The verification of a property is equivalent to checking if the property is satisfied in the model corresponding to the system under the fairness and starting conditions. As described by Emerson [6], concurrency can be expressed by the following equation: concurrency = nondeterminism + fairness.

Analogously, an important aspect of situated reasoning is the notion of *commitment*. The commitment condition specifies when and for how long an agent should pursue its desires and under what conditions an agent should give up its commitment. Thus, a commitment condition embodies the balance between reactivity and goal-directedness of an agent in a situated system. An abstract situated system can thus be viewed as a possible worlds branching-time tree structure with commitment and starting conditions. The verification of a property of the situated system is equivalent to checking if the property is satisfied in the model corresponding to the situated system under the commitment and starting conditions. We could therefore express situated reasoning as follows: situated reasoning = chance + choice + commitment. In the next three sections we formalize these notions.

3 Propositional Branching Time BDI Logics

3.1 Syntax

We first define two languages $\mathrm{CTL}_{\mathrm{BDI}}$ and $\mathrm{CTL}^*_{\mathrm{BDI}}$, which are propositional modal logics based on the branching temporal logics CTL and CTL* [6, 8], respectively. The primitives of these languages include a non-empty set Φ of *primitive propositions*; propositional *connectives* $\vee$ and $\neg$; *modal operators* BEL (agent believes), DESIRE (agent desires), and INTEND (agent intends); and *temporal operators* X (next), U (until), F (sometime in the future or eventually), and E (some path in the future or optionally). Other connectives and operators such as $\wedge$, $\supset$, $\equiv$, G(all times in the future or always), A (all paths in the future or inevitably), $\overset{\infty}{\mathsf{F}}$ (infinitely often), and $\overset{\infty}{\mathsf{G}}$ (almost always) can be defined in terms of the above primitives. The last two operators are defined only for $\mathrm{CTL}^*_{\mathrm{BDI}}$.

There are two types of well-formed formulas in these languages: *state formulas* (which are true in a particular world at a particular time point) and *path formulas* (which are true in a particular world along a certain path). State formulas are defined in the standard way as propositional formulas, modal formulas, and their conjunctions and negations. The object of E and A are path formulas. Path formulas for $\mathrm{CTL}^*_{\mathrm{BDI}}$ can be any arbitrary combination of a linear-time temporal formula, containing negation, disjunction, and the linear-time operators X and U. Path formulas of $\mathrm{CTL}_{\mathrm{BDI}}$ are restricted to be primitive linear-time temporal formu-

las, with no negations or disjunctions and no nesting of linear-time temporal operators.

The *length* of a formula ϕ is denoted by $|\phi|$ and is defined recursively as follows [7]: (a) the length of a primitive proposition is zero; and (b) the length of conjunctive, negated, modal, and temporal formulas is one more than the sum of the sizes of their component formulas. The formula ψ is said to be a *subformula* of ϕ if ψ is a substring of ϕ. Let $Sub(\phi)$ be the set of all subformulas of ϕ.

For example, the formula $\neg p \wedge \mathsf{BEL}(p \wedge q)$ has a length of 4. The subformulas of the above formula is given by $\{\neg p \wedge \mathsf{BEL}(p \wedge q)\neg p, \mathsf{BEL}(p \wedge q), p, p \wedge q, q\}$.

3.2 Possible-Worlds Semantics

We define a *structure* M to be a tuple, $M = (W, T, \mathcal{R}, \mathcal{B}, \mathcal{G}, \mathcal{I}, \Phi)$ where T is a set of *time points*; $\mathcal{R} \subseteq T \times T$ is a total binary *temporal accessibility* relation; Φ is the set of primitive propositions; W is a set of *possible worlds*, where each world w is a tuple of the form $(T_w, \mathcal{R}_w, L_w)$ in which $T_w \in T$ is a set of *time points* in w, $\mathcal{R}_w$ is a restriction of $\mathcal{R}$ to the time points T_w, and L_w is the truth assignment function that assigns to each time point in w a set of propositional formulas, i.e., L_w: $T_w \rightarrow 2^{\Phi}$; finally, $\mathcal{B} \subseteq W \times T \times W$ is the *belief accessibility* relation; and $\mathcal{G}$ and $\mathcal{I}$ are *desire* and *intention accessibility* relations, respectively, that are defined in the same way as $\mathcal{B}$.

Sometimes, we shall view the arcs of the time tree as being labeled with a primitive event, i.e., $\mathcal{R} \subseteq T \times E \times T$, where E is the set of primitive events that is added to the structure M.

For the purposes of this paper, we consider only *finite* structures. The size of a finite structure M is given by the size of the different components of the structure. More formally, $|M| = \mathcal{O}(|W| \cdot (|\mathcal{R}| + |\mathcal{B}| + |\mathcal{G}| + |\mathcal{I}|))$. The size of W is equal to the number of worlds and the size of the relations is equal to the number of elements in the relation.

A *fullpath*, $x = (s_0, s_1, ...)$ in w is an infinite sequence of time points such that $(s_i, s_{i+1}) \in \mathcal{R}_w$ for all i. The suffix fullpath $(s_i, s_{i+1}, ...)$ is denoted by x^i. Satisfaction of a state formula ϕ is given with respect to a structure M, a world w and a time point t, denoted by $M, w, t \models \phi$. Satisfaction of path formulas is given with respect to a structure M, world w, and a fullpath x in world w.

$M, w, t \models \mathsf{BEL}(\phi)$ iff $M, w', t \models \phi$ for all w' satisfying $(w, t, w') \in \mathcal{B}$;
$M, w, t \models \mathsf{E}(\xi)$ iff $M, w, x \models \xi$ where x is a fullpath in world w starting at t;
$M, w, x \models \mathsf{X}(\phi)$ iff $M, w, x^1 \models \phi$;
$M, w, x \models \phi_1 \mathsf{U} \phi_2$ iff for some $i \geq 0$, $M, w, x^i \models \phi_2$ and for all $0 \leq j < i$, $M, w, x^j \models \phi_1$;

The semantics of primitive propositions, negations, and conjunctions of formulas are defined in a standard manner. Desires and intentions are defined as for beliefs but with respect to their corresponding accessibility relations. The relationships between beliefs, desires, and intentions impose different restrictions on these accessibility relations. These relationships have been discussed by us elsewhere [16, 18]. Finally, using the above basic modal operators we can define the additional modal operators as follows: F as *true* $\mathsf{U}\ \phi$; G as $\neg\mathsf{F}(\neg\phi)$; A as $\neg\mathsf{E}(\neg\xi)$; $\overset{\infty}{\mathsf{F}}$ as $\mathsf{GF}(\phi)$; and $\overset{\infty}{\mathsf{G}}$ as $\mathsf{FG}(\phi)$.

3.3 Commitment

Commitment plays an important role in situated reasoning. In a continuously changing environment, commitment lends a certain sense of stability to the reasoning process of an agent. For example, if John is committed to going to the bank at 2 PM, he is unlikely to re-evaluate this decision at every clock tick; instead, he would probably re-evaluate his decision (or give up his commitment) only if there were a significant change in circumstances. In other words, commitment and its relative stability with respect to changes in beliefs results in savings in computational effort and hence a better overall performance [1, 13, 18].

A commitment usually has two parts to it: one is the condition that the agent is committed to maintain, called the *commitment condition*, and the second is the condition under which the agent gives up the commitment, called the *termination condition*. Within this basic framework one can express a number of different types of commitment in the language $\mathrm{CTL_{BDI}}$.

One can define a *strong* or *weak* form of commitment based on whether the agent commits to the commitment condition in *all future paths* or in *some future path*, respectively. More formally, we introduce two operators $\mathsf{C_A}$ and $\mathsf{C_E}$, which are defined as follows:

$\phi_1 \mathsf{C_A} \phi_2 \equiv \phi_1 \supset \mathsf{A}(\phi_1\ \mathsf{U}\ \phi_2)$;
$\phi_1 \mathsf{C_E} \phi_2 \equiv \phi_1 \supset \mathsf{E}(\phi_1\ \mathsf{U}\ \phi_2)$;

where ϕ_1 is a commitment condition and ϕ_2 is a termination condition. Note that ϕ_1 and ϕ_2 can be arbitrary state formulas of the language $\mathrm{CTL_{BDI}}$ and the commitment formulas also belong to $\mathrm{CTL_{BDI}}$. The formula $\phi_1 \mathsf{C_A} \phi_2$ (read as ϕ_1 is inevitably committed until ϕ_2) states that, if the commitment condition ϕ_1 is true, in all future paths the agent will commit to (or maintain) ϕ_1 until the termination condition ϕ_2 is true. Similarly, $\phi_1 \mathsf{C_E} \phi_2$ (read as ϕ_1 is optionally committed until ϕ_2) states that, if ϕ_1 is true, in at least one future path the agent will commit to (or maintain) ϕ_1 until the termination condition ϕ_2 is true.

As the agent has no direct control over its beliefs and desires, there is no way that it can adopt or effectively realize a commitment strategy over these attitudes. However, an agent can choose what to do with its intentions. Thus, we restrict the commitment condition to be of the form $\mathsf{INTEND}(\mathsf{A}\xi)$ or $\mathsf{INTEND}(\mathsf{E}\xi)$. The former we call *full* commitment and the latter a *partial* commitment. The exact form of the termination condition yields different types of commitment. We review three types of commitment that were described elsewhere [18]: *blind commitment* in which the termination condition ϕ_2 is of the form $\mathsf{BEL}(\phi)$; *single-minded commitment* in which ϕ_2 is of the form $\mathsf{BEL}(\phi) \vee \neg\mathsf{BEL}(\mathsf{EF}(\phi))$; and *open-minded commitment* in which ϕ_2 is of the form $\mathsf{BEL}(\phi) \vee \neg\mathsf{DESIRE}(\mathsf{EF}(\phi))$.

An agent that is blindly committed will give up its commitment only when it believes in ϕ, where ϕ is usually a proposition that the agent is striving to achieve. In addition to this, an agent who is single-mindedly committed will give up its commitment when it no longer believes that there exists an option sometime in the future of satisfying the proposition. An agent that is open-mindedly committed will give up its commitment either when it believes in the proposition or when it no longer has the desire to eventually achieve the proposition.

One can combine the above forms of commitment in various ways. For example, the formula $\mathsf{INTEND}(\mathsf{AF}p)\mathsf{C_A}\mathsf{BEL}(p)$ denotes an agent that is blindly and fully committed to achieving p until it believes in p. Similarly, the formula $\mathsf{INTEND}(\mathsf{EF}p)\mathsf{C_A}(\mathsf{BEL}(p) \vee \neg\mathsf{BEL}(\mathsf{EF}p))$ is an example of an agent that is single-mindedly fully committed to achieving p (more intuitively, has decided not to rule out the option of achieving p).

Now one can express the following properties with respect to commitment.

$$\models \phi_1 \mathsf{C_A} \phi_2 \supset (\phi_1 \supset \mathsf{AF}\phi_2);$$
$$\models \phi_1 \mathsf{C_E} \phi_2 \supset (\phi_1 \supset \mathsf{EF}\phi_2);$$
$$\models (\phi_1 \wedge \phi_2)\mathsf{C_A}\phi_3 \equiv (\phi_1 \mathsf{C_A}\phi_3 \vee \neg\phi_2) \wedge (\phi_2 \mathsf{C_A}\phi_3 \vee \neg\phi_1);$$
$$\models \phi_1 \mathsf{C_E}(\phi_2 \vee \phi_3) \equiv (\phi_1 \mathsf{C_E}\phi_2 \vee \phi_1 \mathsf{C_E}\phi_3).$$

The first property reduces commitment to eventuality formulas; whereas the second and third reduce conjunctive and disjunctive conditions of commitment into their simpler forms.

4 Verification of Situated Programs

Our interest is in determining what properties hold of a given agent, in a given environment, under certain initial conditions and under certain commitment conditions. For example, given a robot that is programmed to single-mindedly commit to a certain set of intentions, we may need to prove that, in a particular environment and under particular initial conditions, it will never harm a human.

Given some specification of the agent and the environment, we can generate the branching tree structure corresponding to all possible evolutions of that agent in that environment. [1] This structure represents the model M of the agent and its environment. We assume that the initial environment-agent configuration of the system is given by a state formula ϕ_{START}. We shall refer to the tuple (M, ϕ_{START}) as an *abstract situated system*. As designers of situated reasoning systems we want to be able to verify that given an abstract situated system, certain properties of the system, expressed as state formulas, are true. The abstract situated system and the properties can be expressed in either $\mathrm{CTL_{BDI}}$ or $\mathrm{CTL^*_{BDI}}$. More formally,

[1] We do not address this process in this paper. Methods for generating models used in concurrency theory [6] can be extended for this purpose.

Proposition 1 Verification of abstract situated systems
$(M, \phi_{START}) \models \phi$ iff $\forall$ w, t in M such that $M, w, t \models \phi_{START}$ we have $M, w, t \models \phi$.

Hence, the verification problem for $\mathrm{CTL_{BDI}}$ reduces to the *Model Checking Problem for* $\mathrm{CTL_{BDI}}$ (MCP), defined as follows: Given a structure $M = (W, T, \mathcal{R}, \mathcal{B}, \mathcal{G}, \mathcal{I}, \Phi)$ and a state formula ϕ, determine for each world w and time point t whether $M, w, t \models \phi$.

Informally, an algorithm, AMCP for solving the Model Checking Problem can be given as follows: Start with subformulas of ϕ that are of length 0, determine the worlds and time points where they are true, and then progressively process subformulas of length greater than 0. After i such steps where $|\phi| \leq i$, the set of worlds and time points where ϕ and all its subformulas are true will be known.

This algorithm is a modification of the algorithm given by Emerson and Lei [7] for their Fair Computation Tree Logic (FCTL). The main difference in model checking is the presence of multiple possible worlds. The complexity of the algorithm AMCP is stated below; its details can be found elsewhere [20].

Theorem 1 *Algorithm AMCP correctly solves MCP by labeling each world w and time point t of the structure $M = (W, T, \mathcal{R}, \mathcal{B}, \mathcal{G}, \mathcal{I}, \Phi)$ with the set of subformulas of ϕ true at w and t, and takes $\mathcal{O}(|\phi| \cdot |M|)$ time to run.*

Proof: Analyzing the complexity of the algorithm, we obtain the following relation for the time taken by the algorithm AMCP (denoted by $Time_{AMCP}$).

$$Time_{AMCP}(M, \phi)$$
$$\leq \sum_{\alpha \in Sub(\phi)} \sum_{i=1}^{|W|} \mathcal{O}(|T|) + \mathcal{O}(|\mathcal{B}|) + \mathcal{O}(|\mathcal{G}|) + \mathcal{O}(|\mathcal{I}|) + \mathcal{O}(|\mathcal{R}|)$$
$$\leq \mathcal{O}(|\phi| \cdot |M|). \ \square$$

Although $\mathrm{CTL_{BDI}}$ can capture different forms of commitment, it is still not expressive enough for our purposes. In particular, we cannot state that in all paths that satisfy a certain commitment formula, say ξ_1, a property, say ξ_2, holds. More formally, we cannot express $\mathsf{A}[\xi_1 \supset \xi_2]$. For this we need to examine the verification of $\mathrm{CTL^*_{BDI}}$-based situated programs.

The language $\mathrm{CTL^*_{BDI}}$ subsumes the language CTL^*, which in turn subsumes the linear-time temporal language LTL. Hence, the complexity of model checking for $\mathrm{CTL^*_{BDI}}$ has to be the same or greater than that of the model checking for LTL. It has been shown [15] that the complexity of model checking in LTL is linear in the size of the structure and exponential in the size of the given formula. Therefore, we have the following result:

Theorem 2 *The time taken for model checking in* $\mathrm{CTL^*_{BDI}}$ *is greater than or equal to $\mathcal{O}(|M| \cdot 2^{|\phi|})$.*

An algorithm with this degree of complexity is likely to prove intractable for practical use. We therefore need a language that is more expressive than $\mathrm{CTL_{BDI}}$ but whose model checking complexity is less than that of $\mathrm{CTL^*_{BDI}}$. We show how this can be accomplished in the next section.

5 Committed CTL_{BDI}

We define a language $CCTL_{BDI}$ (Committed CTL_{BDI}) in which a state formula is essentially a CTL_{BDI} state formula except that we now allow modalities of the form A_ξ ("for all *committed* paths ξ") and E_ξ ("for some *committed* path ξ"). The formula $A_\xi \xi_1$ can be viewed as a CTL^*_{BDI} formula $A(\xi \supset \xi_1)$ and similarly $E_\xi \xi_1$ can be viewed as a CTL^*_{BDI} formula $E(\xi \wedge \xi_1)$. Hence, the semantics of $CCTL_{BDI}$ is the same as the semantics of CTL^*_{BDI}. We shall call the above conversion **-conversion* and transform a formula ϕ of $CCTL_{BDI}$ as a transformed formula ϕ^* of CTL^*_{BDI}.

However, if the commitment formula ξ is any unrestricted path formula of CTL^*_{BDI} the complexity of model-checking in $CCTL_{BDI}$ can be shown to be the same as that of CTL^*_{BDI}. To reduce the exponential time complexity of model-checking in CTL^*_{BDI} and to improve on the expressive power of CTL_{BDI}, we need to restict the path formula ξ in A_ξ and E_ξ .

In the literature on reasoning about concurrent programs, Emerson and Lei [7] define a *fairness constraint* whose canonical form is given by $\bigwedge_{i=1}^{n}(\overset{\infty}{F} p_i \vee \overset{\infty}{G} q_i)$. This canonical form is sufficiently powerful to express many different types of fairness conditions and at the same time has a polynomial time complexity for model checking.

As p_i and q_i are restricted to propositional formulas, the above canonical form is not very useful for expressing different types of commitment. In fact, we want to check the satisfiability of formulas where the commitment formula α is satisfied infinitely often, i.e. $\overset{\infty}{F}(\alpha)$, where α can be either $\phi C_A \psi$ or $\phi C_E \psi$, for state formulas ϕ and ψ of CTL_{BDI}. Hence, we define our *commitment constraint*, Ξ, to be the canonical form $\bigwedge_{i=1}^{n}(\overset{\infty}{F}(\alpha_i) \vee \overset{\infty}{G}(\beta_i))$, where α_i and β_i are commitment formulas of the form $\phi_i C_A \psi_i$ or $\phi_i C_E \psi_i$. Note that other forms of commitment constraints such as $\phi_1 U \phi_2$ or $FA(\phi_1 U \phi_2)$, although simpler than our canonical form, have a greater computational complexity. checking. We use $M, w, t \models_\Xi \phi$ as an abbreviation for $M, w, t \models \phi^*$, where ϕ^* is a CTL^*_{BDI} formula obtained from ϕ using the *-conversion. An abstract situated system based on $CCTL_{BDI}$ is given by the tuple (M, ϕ_{START}, Ξ).

With this framework we extend the results of Emerson and Lei [7], given for FCTL, to our committed CTL_{BDI} logic. The extensions to FCTL are twofold: (i) the introduction of possible worlds extends the expressive power of CTL and results in a complex structure on which to perform model checking; and (ii) the commitment constraint is more complex involving modal operators and path quantifiers.

We first need some definitions. A fullpath x is a *committed path* in structure M and world w under the commitment assumption Ξ if $M, w, x \models \Xi$ holds. A time point t is a *committed state* in world w iff, starting from t, there is some committed path in world w. A world w is a *committed world* in structure M iff there is a committed path in structure M in w. A substructure c of a world w is called a *committed component* if c is a total, strongly connected component of w that contains some committed path. Now we state the verification of $CCTL_{BDI}$ based situated systems.

Proposition 2 Verification of abstract situated systems – *$(M, \phi_{START}, \Xi) \models \phi$ iff $\forall$ w , t in M such that $M, w, t \models \phi_{START}$ we have $M, w, t \models_\Xi \phi$.*

We define the *Model Checking Problem for Committed* CTL_{BDI} (CMCP) as follows: Given a structure M, a commitment constraint Ξ, and a state formula ϕ, determine for each world w and time point t whether $M, w, t \models_\Xi \phi$. We also define the *Model Checking Problem for Committed States* (CSP) as follows: Given a structure $M = (W, T, \mathcal{R}, \mathcal{B}, \mathcal{G}, \mathcal{I}, \Phi)$ and a commitment constraint Ξ, determine for each world w and time point t whether there is a fullpath x in M starting at t such that $M, w, x \models \Xi$.

In the following, we use techniques similar to that of Emerson and Lei [7] to reduce CMCP to CSP. The reduction relies on the nature of the commitment constraint Ξ. In particular, the fact that $\overset{\infty}{F}$ and $\overset{\infty}{G}$ are oblivious to the addition and deletion of finite prefixes is used to reduce CMCP to CSP. Also, formulas of the form $E_\xi X \phi$, $E_\xi[\phi U \psi]$, $E_\xi[\neg(\phi U \psi)]$, $\phi C_A \psi$, and $\phi C_E \psi$ can be reduced to model checking of primitive propositions, formulas of the form $E_\xi X$ *true*, and $E_\xi G(\psi)$ [20].

The algorithm for model checking in $CCTL_{BDI}$, ACMCP, takes as input the structure M, a $CCTL_{BDI}$ formula ϕ, and a commitment constraint Ξ, and results in the truth assignment of ϕ and all its subformulas. The algorithm ACMCP uses algorithms ACSP and ACC and operates in a similar fashion to the algorithm AMCP. However, ACMCP computes all the committed worlds and the committed states in each world using ACSP, adds the formula $E_\xi X$ *true* to all such committed states, and checks all temporal formulas against such committed states.

Given a structure M and a commitment constraint Ξ, the algorithm ACSP outputs a set of committed worlds and the set of committed states in each one of these committed worlds. The algorithm operates by first computing all the strongly connected components of a world and then checking, for each one of these components, if it is a committed component or not by calling the algorithm ACC. The set of committed states for a given world is a union of all the time points of all the committed components and all the time points that can reach these. The process is repeated for each and every world in the structure M.

Given a structure M, a strongly connected world c, and a commitment constraint Ξ, the algorithm ACC determines if c is a committed component or not. Unlike the algorithm for FCTL [7], in which the object of the operator $\overset{\infty}{F}$ was a propositional formula, we have a specific CTL_{BDI} formula. Hence, for each conjunct, we first run the model checking algorithm AMCP to determine the worlds and time points at which the

commitment formula α_i is true. If α_i is true at all time points of the given strongly connected world c then c is a committed component. However, if one of the α_i's is false at a time point we create a new connected structure c' containing all time points where β_i is true and recursively check if the new connected structure is a committed component with a modified structure M', and a modified commitment constraint Ξ'.

The details of the algorithms ACMCP, ACSP, and ACC are given elsewhere [20]. We analyze their complexity below.

Proposition 3 *Given a structure M, a strongly connected world $c = (T_c, \mathcal{R}_c, L_c)$, where T_c is finite, and a commitment constraint Ξ the algorithm ACC decides whether c is a committed component w.r.t. Ξ in time $\mathcal{O}(|M| \cdot |\Xi|^2)$.*

Proof: The time taken for the algorithm ACC, say $Time_{ACC}$, is a function of M, Ξ, and the number of conjuncts in Ξ, say k. Let $D_1, ..., D_l$ be the strongly connected components of c' and d_i the size of D_i. We can now write down the following recurrence relation:

$$
\begin{aligned}
& Time_{ACC}(|M|, |\Xi|, k) \\
& \leq \textstyle\sum_{j=1}^{k} \mathcal{O}(Time_{AMCP}(M, \alpha_i)) + \\
& \quad \textstyle\sum_{j=1} Time_{ACC}(d_j, |\Xi'|, k\text{-}1) \\
& \leq \textstyle\sum_{j=1}^{k} \mathcal{O}(|M| \cdot |\alpha_i|)) + \\
& \quad \textstyle\sum_{j=1} Time_{ACC}(d_j, |\Xi'|, k\text{-}1) \\
& \leq \mathcal{O}(|M| \cdot |\Xi|) + Time_{ACC}(m, n, k\text{-}1) \\
& \leq \mathcal{O}(|M| \cdot |\Xi|) + ... + \mathcal{O}(|M| \cdot |\Xi|) \text{ [}k\text{ times]} \\
& \leq \mathcal{O}(|M| \cdot |\Xi| \cdot k) \\
& \leq \mathcal{O}(|M| \cdot |\Xi|^2).
\end{aligned}
$$

Proposition 4 *The algorithm ACSP(M, Ξ) is an algorithm for CSP of time complexity $\mathcal{O}(|W| \cdot |M| \cdot |\Xi|^2)$.*

Proof: The time taken for the algorithm ACSP, say $Time_{ACSP}$, is a function of M and Ξ. Let $c_1, ..., c_l$ be the strongly connected components of w. We can now write down the following recurrence relation:

$$
\begin{aligned}
& Time_{ACSP}(M, \Xi) \\
& \leq \textstyle\sum_{i=1}^{|W|} ((\sum_{j=1}^{l} Time_{ACC}(M', c_j, \Xi) + \\
& \qquad \mathcal{O}(|c_j|)) + \mathcal{O}(|\mathcal{R}_w|)) \\
& \leq \textstyle\sum_{i=1}^{|W|} (\mathcal{O}(|M| \cdot |\Xi|^2) + \mathcal{O}(|\mathcal{R}_w|)) \\
& \leq \mathcal{O}(|W| \cdot |M| \cdot |\Xi|^2)
\end{aligned}
$$

Theorem 3 *Solving the model checking problem for committed branching temporal BDI logic,* CCTL_{BDI} *will take $\mathcal{O}(|\phi| \cdot |W| \cdot |M| \cdot |\Xi|^2)$ time to run.*

Proof: We know that the complexity of the algorithm AMCP is $\mathcal{O}(|\phi| \cdot |M|)$. Algorithm ACMCP works exactly as AMCP except that it operates on committed states. Hence, the time taken for ACMCP can be given as follows:

$$
\begin{aligned}
Time_{ACMCP} & \leq \mathcal{O}(|\phi| \cdot \max(|M|, Time_{ACSP}(M,\Xi))); \\
& \leq \mathcal{O}(|\phi| \cdot \max(|M|, (|W| \cdot |M| \cdot |\Xi|^2))); \\
& \leq \mathcal{O}(|\phi| \cdot |W| \cdot |M| \cdot |\Xi|^2).
\end{aligned}
$$

In FCTL [7] the algorithm for computing connected components is $\mathcal{O}(|c| \cdot |\Xi|^2)$, for computing fair states is $\mathcal{O}(|M| \cdot |\Xi|^2)$, and for model checking is $\mathcal{O}(|\phi| \cdot |M| \cdot |\Xi|^2)$. However, in our case the algorithm ACC is linear in the size of the entire structure M because of the presence of modal formulas (i.e., belief, desire, and intention formulas) in the commitment constraint. The increased complexity of computing fair states and model checking is due to the presence of multiple possible worlds.

6 Example

Consider a robot that can perform two tasks, each involving two actions. For the first task, the robot can go to the refrigerator and take out a can of beer (denoted by `gf`) and bring it to the living room (`bb`). For the second task, the robot can go to the main door of the house (`gd`) and open the door (`od`). The only uncertainty in the environment is the presence or absence of a beer can in the refrigerator. For simplicity, we assume that the act of going to the refrigerator also involves opening the door and checking for the can of beer. If there is no can in the refrigerator the act `gf` is said to have failed and the next act of bringing beer cannot be performed. We assume that all other acts succeed when executed.

Given appropriate specifications of such a robot and its environment and the commitment constraint as designers of these robots, we need to guarantee that the robot satisfies certain properties. For example, we need to guarantee that (a) when the robot has a desire to serve beer it will inevitably eventually serve beer; and (b) when the robot has a desire to serve beer and a desire to answer the door, and there is beer in the fridge, it will inevitably eventually realize both desires, rather than shifting from one task to the other without completing either of them. [2]

We consider two model structures M_1 and M_2. First, we start by specifying directly the external model structure M_1. Generation of the external model structure from the agent and environment specifications is beyond the scope of this paper. A partial description of the structure M_1 is shown in Figure 1. World `w1` depicts the alternatives available to the robot when it has to perform both the tasks and the environment is such that there is a beer can in the refrigerator. The dotted lines refer to additional future paths, that can be described in an analogous manner. One can view worlds `w2` and `w3` as world `w1` after the agent has executed the act of either going to the refrigerator or going towards the door, respectively. Similarly, `w4` and `w5` are evolutions of `w2`; `w6` and `w7` are evolutions of `w3`.

We introduce three propositions:
`beer-in-refrigerator`,

[2] Note that real-time systems may require time bounds to be specified on these properties. For example, we may want to guarantee that beer will be served to the guests within ten minutes of their arrival. However, the logics presented in this paper are not powerful enough to reason about such quantitative properties.

served-beer and **answered-door**. The proposition **beer-in-refrigerator** is true at all times in the worlds **w1-w7**. The proposition **served-beer** will be true in worlds **w1-w7** after the act of bringing the beer (**bb**) and the proposition **answered-door** will be true in all worlds after the act of opening the door **od**.

Next we examine the belief, desire, intention relations of the agent. The world **w1** of Figure 1 shows the various time points. The belief relations for world **w1** at various time points are given as follows: (**w1**, **t1**, **w1**), (**w1**, **t2**, **w2**), (**w1**, **t3**, **w3**), (**w1**, **t4**, **w4**), (**w1**, **t5**, **w5**), (**w1**, **t6**, **w6**), (**w1**, **t7**, **w7**),... $\in \mathcal{B}$. Desire and intention relations can be defined similarly. Further, we assume that the belief relations do not change when actions are performed. In other words, we also have (**w2**, **t2**, **w2**), (**w2**, **t4**, **w4**), (**w2**, **t5**, **w5**), ... $\in \mathcal{B}$. Similar relationships hold for worlds **w3--w7**. This completes our description of the structure M_1.

Consider a starting state in which the robot believes that there is beer in the refrigerator and has the intention to inevitably eventually have served beer and to inevitably eventually have answered the door. More formally, ϕ_{START} is:

$$\begin{aligned}&\mathsf{BEL}(\texttt{beer-in-refrigerator}) \wedge\\ &\mathsf{INTEND}(\mathsf{AF}(\texttt{served-beer})) \wedge\\ &\mathsf{INTEND}(\mathsf{AF}(\texttt{answered-door})).\end{aligned}$$

We consider four instances of the commitment constraint; the first two instances are blind commitments towards intentions and the last two are single-minded commitments towards intentions. Also, the commitment in the first and third instances are only towards serving the beer, while the commitments in the second and fourth instances are towards serving the beer and answering the call. More formally,

$$\begin{aligned}\Xi_1 \equiv\ & \overset{\infty}{\mathsf{F}}(\mathsf{INTEND}(\mathsf{AF}(\texttt{served-beer}))\mathsf{C_A}\\ & \mathsf{BEL}(\texttt{served-beer}));\end{aligned}$$

$$\begin{aligned}\Xi_2 \equiv\ & \Xi_1 \wedge \overset{\infty}{\mathsf{F}}(\mathsf{INTEND}(\mathsf{AF}(\texttt{answered-door}))\mathsf{C_A}\\ & \mathsf{BEL}(\texttt{answered-door})));\end{aligned}$$

$$\begin{aligned}\Xi_1 \equiv \Xi_3 \equiv\ & \overset{\infty}{\mathsf{F}}(\mathsf{INTEND}(\mathsf{AF}(\texttt{served-beer}))\mathsf{C_A}\\ & (\mathsf{BEL}(\texttt{served-beer}) \vee\\ & \neg\mathsf{BEL}(\mathsf{EF}(\texttt{served-beer}))));\end{aligned}$$

$$\begin{aligned}\Xi_4 \equiv \Xi_3 \wedge\ & \overset{\infty}{\mathsf{F}}(\ \mathsf{INTEND}(\mathsf{AF}(\texttt{answered-door}))\mathsf{C_A}\\ & (\mathsf{BEL}(\texttt{answered-door}) \vee\\ & \neg\mathsf{BEL}(\mathsf{EF}(\texttt{answered-door})))).\end{aligned}$$

Using Proposition 2 and algorithm ACMCP we can show the following:

$$\begin{aligned}&(M_1, \phi_{START}, \Xi_1) \models \mathsf{AF}(\texttt{served-beer});\\ &(M_1, \phi_{START}, \Xi_2) \models \mathsf{AF}(\texttt{served-beer}) \wedge\\ &\qquad\qquad\qquad\quad \mathsf{AF}(\texttt{answered-door});\\ &(M_1, \phi_{START}, \Xi_3) \models \mathsf{AF}(\texttt{served-beer});\\ &(M_1, \phi_{START}, \Xi_4) \models \mathsf{AF}(\texttt{served-beer}) \wedge\\ &\qquad\qquad\qquad\quad \mathsf{AF}(\texttt{answered-door}).\end{aligned}$$

Next, consider the structure M_2 which consists of worlds **w1-w7** shown in Figure 6 and additional worlds where the proposition **beer-in-refrigerator** is false at all time points. Transitions between these worlds are similar to worlds **w1-w7** except that the act **gf** fails (as there is no beer can in the refrigerator) and is followed by the act of going to the main door, namely **gd**, rather than the act of bringing the beer, namely **bb**.

Let us consider two robots, named **Mark I** and **Mark II**.[3] Intuitively, **Mark I** robot does not change its belief about there being beer in the refrigerator at some time point in the future, even if it notices at this time point that there is no beer in the refrigerator. On the other hand, **Mark II** robot changes its belief about the beer being in the refrigerator as soon as it notices that there is none.

With the structure M_2 we can show that a single-mindedly committed **Mark II** agent will drop its commitment to maintain the intention of inevitably eventually serving beer. On the other hand, a single-mindedly committed **Mark I** agent will maintain this commitment forever. More formally, we can show the following:

$$\begin{aligned}&(M_2, \phi_{START}, \Xi_3)\\ &\quad\models\ \neg\mathsf{AF}(\texttt{served-beer}) \wedge\\ &\qquad\ \mathsf{AG}(\mathsf{INTEND}(\mathrm{I}, \mathsf{AF}(\texttt{served-beer}))) \wedge\\ &\qquad\ \mathsf{BEL}(\mathrm{I}, \mathsf{AF}(\texttt{served-beer}));\\ &(M_2, \phi_{START}, \Xi_3)\\ &\quad\models\ \neg\mathsf{AF}(\texttt{served-beer}) \wedge\\ &\qquad\ \neg\mathsf{AG}(\mathsf{INTEND}(\mathrm{II}, \mathsf{AF}(\texttt{served-beer}))) \wedge\\ &\qquad\ \mathsf{BEL}(\mathrm{II}, \neg\mathsf{AF}(\texttt{served-beer})).\end{aligned}$$

In summary, we have considered two different model structures, one where the robot completes both its tasks, the second where it is impossible for the robot to complete one of the tasks, but yet one of the robots maintains its commitment to this task forever, while the other robot reconciles itself to the impossibility of completing the task and gives it up. The purpose of this exercise has been to show how global properties of situated systems can be verified under a variety of rational behaviours obtained by varying the model structure M and the commitment constraint Ξ.

7 Comparisons and Conclusions

Cohen and Levesque [4] describe agents by adopting a possible worlds structure in which each world is a linear-time temporal structure and consider fanatical and relativized forms of commitment. A fanatical commitment is similar to our definition of a single-minded agent committed to its intention, i.e., $\mathsf{INTEND}(\mathsf{AF}\phi)\mathsf{C_A}(\mathsf{BEL}(\phi) \vee \mathsf{BEL}(\mathsf{AG}\neg\phi))$. A relativized commitment is one in which the agent has a persistent intention towards a proposition until it believes in the proposition or until some other proposition is believed. This can be expressed as

[3] Although, we have not described a multi-agent $\mathrm{CTL_{BDI}}$ logic the modifications required to do so are straightforward. Also, as long as we do not introduce common knowledge operators, the complexity of model checking in such multi-agent modal logics will be of the same order as single-agent modal logics (See Halpern and Moses [11] for some results in model-checking in multi-agent belief logics)

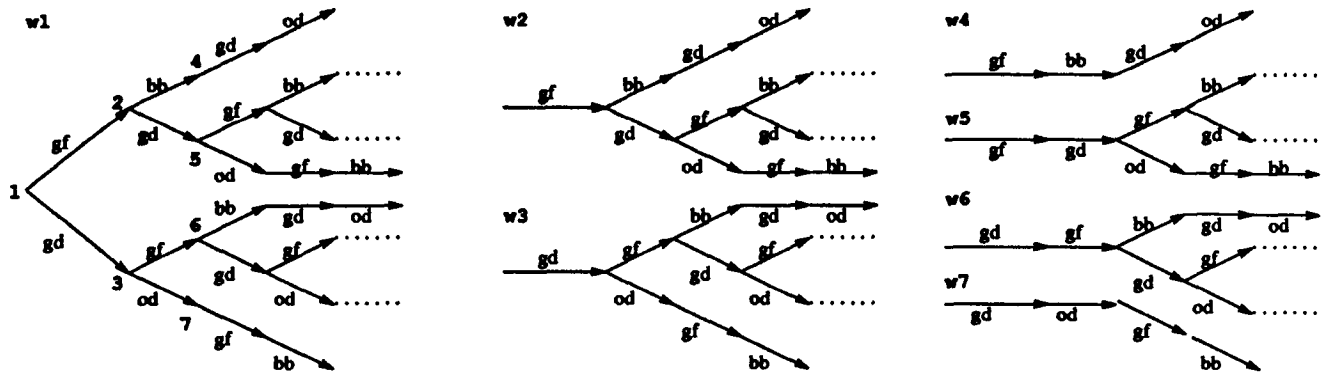

Figure 1: A partial structure of M_1

INTEND(AFϕ)C$_A$(BEL(ϕ) $\vee$ BEL(AG$\neg\phi$) $\vee$ BEL(ψ)). Cohen and Levesque do not address the issue of model checking in their logic. However, as their logic subsumes linear-time temporal logic (LTL), the process of model checking in their logic will be at least as hard as the model checking for LTL; namely, linear in the size of the structure and exponential in the size of the given formula [15].

Singh [24] presents a branching-time intention logic based on *CTL**. Various rationality postulates relating to beliefs, intentions, and actions are analyzed. Also, like Cohen and Levesque, Singh uses his logic only as a specification to characterize different behaviours and does not provide any guidelines for the design or verification of such rational agents. Shoham's work [23] spans both theory and language design, but does not address the issue of verification either.

This paper goes beyond this earlier work and provides a methodology for formally verifying properties of rational situated systems. Starting from a reasonably rich model structure, we have described three propositional logics and analyzed their relative expressive power. Furthermore, the linear time and polynomial time complexity of two of these algorithms makes them potentially useful for verifying practical situated systems.

Our work draws its inspiration from the field of concurrency theory [6], especially that field's contribution to the techniques of model checking. We extend the results of Emerson and Lei [7] by showing that the linear time and polynomial time complexities of model checking hold for logics more expressive than CTL and Fair CTL logics. Also, the complexities are not greatly affected by the number of different modalities – the complexity seems to be dependent on the underlying temporal structure. More importantly, this paper demonstrates the generality of the model-checking technique [12] and extends it to a new domain; namely, the verification of situated reasoning systems. The close correspondence between fairness and commitment, and concurrent programs and situated systems, lays a strong theoretical foundation for the design and verification of situated reasoning systems.

However, a number of open problems with respect to this approach remain. First, we need to address the process of *model generation* whereby, given an agent specification and/or environment specification, the appropriate model structure is automatically generated. Second, we have used model checking as a means of verifying global properties, i.e., from an external observer viewpoint. Similar techniques can be used by the agent internally. In this case, we may want to build the model incrementally, rather than assuming that the entire model structure is given to us. Third, the size of the structures we are dealing with are likely to be quite large and techniques to reduce this would be valuable.

Although a number of issues in the model-theoretic design and verification of situated systems are yet to be resolved, our work indicates, for the first time, that the expressive multi-modal, branching-time logics can possibly be used in practice to verify the properties of situated reasoning systems.

References

[1] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, Massachusetts, 1987.

[2] M. E. Bratman, D. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, pages 14–23, 1987.

[4] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[5] J. Doyle. Rationality and its roles in reasoning. *Computational Intelligence*, 8(2):376–409, 1992.

[6] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics*, pages 995–1072. Elsevier Science Publishers and The MIT Press, Amsterdam and Cambridge, USA, 1990.

[7] E. A. Emerson and C-L Lei. Modalities for Model Checking: Branching Time Logic Strikes Back. *Science of Computer Programming*, 8:275–306, 1987.

[8] E. A. Emerson and J. Srinivasan. Branching time temporal logic. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 123–172. Springer-Verlag, Berlin, 1989.

[9] N. Francez and D. Kozen. Generalized fair termination. In *Proceedings of the 11th Annual ACM Symposium on Principles of Programming Languages*, pages 46–53, 1984.

[10] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proceedings of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398, 1986.

[11] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[12] J. Y. Halpern and M. Y. Vardi. Model checking vs. theorem proving: A manifesto. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 325–334. Morgan Kaufmann Publishers, San Mateo, 1991.

[13] D. Kinny and M. P. Georgeff. Commitment and effectiveness of situated agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, 1991.

[14] L. Lamport. Sometimes is sometimes "not never" – on the temporal logic of programs. In *Proceedings of the 7th Annual ACM Symposium on Principles of Programming Languages*, pages 174–185, 1980.

[15] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages*, pages 97–107, 1985.

[16] A. S. Rao and M. P. Georgeff. Asymmetry thesis and side-effect problems in linear time and branching time intention logics. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, 1991.

[17] A. S. Rao and M. P. Georgeff. Deliberation and its role in the formation of intentions. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (UAI-91)*, 1991.

[18] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, San Mateo, 1991.

[19] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[20] A. S. Rao and M. P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. Technical Report 37, Australian Artificial Intelligence Institute, Carlton, Australia, 1992.

[21] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, San Mateo, California, 1986. Morgan Kaufmann Publishers, Inc.

[22] S. Russell and S. Subramanian. On provably ralphs. In E. Baum, editor, *Computational Learning and Cognition: The Third NEC Research Symposium*. SIAM Press, 1992.

[23] Y. Shoham. Agent0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 704–709, 1991.

[24] M. P. Singh. A logic of situated know-how. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 343–348, 1991.