Tractable Multiagent Planning for Epistemic Goals

Wiebe van der Hoek
Department of Computer Science
Utrecht University & University of Liverpool
The Netherlands & U.K.
wiebe@cs.uu.nl

Michael Wooldridge
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, U.K.
M.J.Wooldridge@csc.liv.ac.uk

ABSTRACT

An epistemic goal is a goal about the knowledge possessed by an agent or group of agents. In this paper, we address the problem of how plans might be developed for a group of agents to cooperate to bring about such a goal. We present a novel approach to this problem, in which the problem is formulated as one of *model checking* in Alternating Temporal Epistemic Logic (ATEL). After introducing this logic, we present a model checking algorithm for it, and show that the model checking problem for this logic is tractable. We then show how multiagent planning can be treated as a model checking problem in ATEL, and discuss the related issue of checking knowledge preconditions for multiagent plans. We illustrate the approach with an example. We then describe how this example was implemented using the MOCHA model checking system, and conclude by discussing the relationship of our work with that of others in the planning and speech acts communities.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Models—*modal logic, temporal logic, model checking*

General Terms

Theory, Verification

Keywords

Epistemic & Temporal Logic, Model Checking, Planning

1. INTRODUCTION

An *epistemic goal* is a goal about the knowledge possessed by an agent or group of agents. For example, Alice may have the goal of making Bob know the combination to the safe; or Alice may have the goal of making it common knowledge between Bob and Chris that it is raining in London. An epistemic goal thus relates to the knowledge (or, more generally, the beliefs) possessed by an agent or group of agents. In this paper, we address the problem of how plans might be developed for a group of agents to cooperate to bring about such goals. In the simplest case, one agent might work in isolation to bring about an epistemic state in an agent; in a slightly more complex case, an agent might work alone to cause a group of agents to collectively know something. Alternatively, a group of agents might attempt to bring about a particular epistemic state in either an individual agent, or, in the most general case, a group of agents. All of these possibilities are covered by our work.

Our approach to this problem is based on the paradigm of *planning as model checking* pioneered by Giunchiglia and colleagues [11]. In this approach, a planning domain is encoded as a semantic structure (model) for a particular logic, and the goal is expressed as a formula asserting that the desired state of affairs is possible to achieve. Planning is then viewed as a process of checking that the formula representing the goal is satisfied in the model representing the domain. Most work in this area has used Computation Tree Logic (CTL) [8]. In our approach, we use a temporal logic that incorporates knowledge operators [19]. This logic is called Alternating Temporal Epistemic Logic (ATEL), and is an extension of the Alternating Temporal Logic (ATL) of Alur, Henzinger, and Kupferman [3]. ATL is a novel generalisation of CTL in which the path quantifiers of CTL are replaced by *cooperation modalities*: the ATL formula $\langle\langle\Gamma\rangle\rangle\Diamond\varphi$, where Γ is a group of agents, expresses the fact that Γ can cooperate to eventually bring about φ. The CTL path quantifiers A ("on all paths…") and E ("on some paths…") can be expressed in ATL by the cooperation modalities $\langle\langle\emptyset\rangle\rangle$ ("the empty-set of agents can cooperate to…") and $\langle\langle\Sigma\rangle\rangle$ ("the grand coalition of all agents can cooperate to…"). ATEL extends ATL by the addition of operators for representing knowledge. As well as operators for representing the knowledge of individual agents, ATEL includes modalities for representing what "everyone knows" and common knowledge [10, 14].

The remainder of the paper is structured as follows. We begin by presenting Alternating Epistemic Transition Systems, the semantic structures that we use to represent out domains. We then introduce the logic ATEL, giving its semantics in terms of these sructures. We present a model checking algorithm for ATEL in section 4, and show that the complexity of the ATEL model checking problem is PTIME-complete. In section 5, we introduce the multiagent planning problem, and show how this problem can be reduced to an ATEL model checking problem. In section 6, we introduce an example scenario; we describe how this scenario is implemented using a freely available ATL model checking system called MOCHA [2, 1], and we show how, using this system, we were able to check various ATEL properties — in particular, we show how we were able to check the existence of multiagent plans for epistemic goals in this scenario. We conclude with some comments and a short discussion on related work.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

2. ALTERNATING EPISTEMIC TRANSITION SYSTEMS

We begin by introducing the semantic structures used to represent our domains. These structures are a straightforward extension of the alternating transition systems used by Alur and colleagues to give a semantics to ATL. Formally, an *alternating epistemic transition system* (AETS) is a tuple

$$\langle \Pi, \Sigma, Q, \sim_1, \ldots, \sim_n, \pi, \delta \rangle, \text{ where}$$

- Π is a finite, non-empty set of *atomic propositions*;
- $\Sigma = \{a_1, \ldots, a_n\}$ is a finite, non-empty set of *agents*;
- Q is a finite, non-empty set of *states*;
- $\sim_a \subseteq Q \times Q$ is an *epistemic accessibility relation* for each agent $a \in \Sigma$ — we require that each $\sim_a$ is an equivalence relation;
- $\pi : Q \to 2^{\Pi}$ gives the set of primitive propositions satisfied in each state;
- $\delta : Q \times \Sigma \to 2^{2^Q}$ is the system transition function, which maps states and agents to the choices available to these agents. Thus $\delta(q, a)$ is the set of choices available to agent a when the system is in state q. We require that this function satisfy the requirement that the system is completely controlled by its component agents: for every state $q \in Q$ and every set $Q_1, \ldots, Q_n$ of choices $Q_i \in \delta(q, a)$, the intersection $Q_1 \cap \cdots \cap Q_n$ is a singleton. This means that if every agent has made his choice, the system is completely determined.

Epistemic Relations. If $\Gamma \subseteq \Sigma$, we denote the union of Γ's accessibility relations by $\sim^E_\Gamma$, so $\sim^E_\Gamma = (\bigcup_{a \in \Gamma} \sim_a)$. Also, $\sim^C_\Gamma$ denote the transitive closure of $\sim^E_\Gamma$. We will later use $\sim^C_\Gamma$ and $\sim^E_\Gamma$ to give a semantics to the common knowledge and "everyone knows" modalities in our logic [10].

Computations. For two states $q, q' \in Q$ and an agent $a \in \Sigma$, we say that state q' is an *a-successor* of q if there exists a set $Q' \in \delta(q, a)$ such that $q' \in Q'$. Intuitively, if q' is an a-successor of q, then q' is a possible outcome of one of the choices available to a when the system is in state q. We denote by $succ(q, a)$ the set of a successors to state q. We say that q' is simply a *successor* of q if for all agents $a \in \Sigma$, we have $q' \in succ(q, a)$; intuitively, if q' is a successor to q, then when the system is in state q, the agents Σ can cooperate to ensure that q' is the next state the system enters.

A *computation* of an AETS $\langle \Pi, \Sigma, Q, \sim_1, \ldots, \sim_n, \pi, \delta \rangle$ is an infinite sequence of states $\lambda = q_0, q_1, \ldots$ such that for all $u > 0$, the state q_u is a successor of q_{u-1}. A computation starting in state q is referred to as a *q-computation*; if $u \in I\!N$, then we denote by $\lambda[u]$ the u'th state in λ; similarly, we denote by $\lambda[0, u]$ and $\lambda[u, \infty]$ the finite prefix $q_0, \ldots, q_u$ and the infinite suffix $q_u, q_{u+1}, \ldots$ of λ respectively.

Strategies and Their Outcomes. Intuitively, a *strategy* is an abstract model of an agents decision-making process; a strategy may be thought of as a kind of plan for an agent. By *following* a strategy, an agent can bring about certain states of affairs. Formally, a strategy f_a for an agent $a \in \Sigma$ is a total function $f_a : Q^+ \to 2^Q$, which must satisfy the constraint that $f_a(\lambda \cdot q) \in \delta(q, a)$ for all $\lambda \in Q^*$ and $q \in Q$. Given a set $\Gamma \subseteq \Sigma$ of agents, and an indexed set of strategies $F_\Gamma = \{f_a \mid a \in \Gamma\}$, one for each agent $a \in \Gamma$, we define $out(q, F_\Gamma)$ to be the set of possible outcomes that may occur if every agent $a \in \Gamma$ follows the corresponding strategy f_a, starting when the system is in state $q \in Q$. That is, the set $out(q, F_\Gamma)$ will contain all possible q-computations that the agents Γ can "enforce" by cooperating and following the strategies in F_Γ. Note that the "grand coalition" of all agents in the system can cooperate to uniquely determine the future state of the system, and so $out(q, F_\Sigma)$ is a singleton. Similarly, the set $out(q, F_\emptyset)$ is the set of all possible q-computations of the system.

3. ALTERNATING TEMPORAL EPISTEMIC LOGIC

Alternating epistemic transition systems are the structures we use to model the systems of interest to us. We now introduce a language to represent and reason about these structures. This language — alternating temporal epistemic logic (ATEL) — is an extension of the alternating temporal logic (ATL) of Alur, Ganzinger, and Kupferman [3], which in turn takes its inspiration from the branching temporal logics CTL and CTL* [8]. Just as formulae of alternating temporal logic are interpreted with respect to alternating transition systems, formulae of ATEL are interpreted with respect to the alternating epistemic transition systems introduced above.

Before presenting the detailed syntax of ATEL, we give an overview of the intuition behind its key constructs. ATEL is an extension of classical propositional logic, and so it contains all the conventional connectives that one would expect to find: $\wedge$ ("and"), $\vee$ ("or"), $\neg$ ("not"), $\to$ ("implies"), and so on. In addition, ATEL contains the temporal cooperation modalities of ATL, as follows. The formula $\langle\langle \Gamma \rangle\rangle \Box \varphi$, where Γ is a group of agents, and φ is a formula of ATEL, means that the agents Γ can work together (cooperate) to ensure that φ is always true. Similarly, $\langle\langle \Gamma \rangle\rangle \bigcirc \varphi$ means that Γ can cooperate to ensure that φ is true in the *next* state. The formula $\langle\langle \Gamma \rangle\rangle \varphi \mathcal{U} \psi$ means that Γ can cooperate to ensure that φ remains true *until* such time as ψ is true — and moreover, ψ, *will* be true at some time in the future.

An ATEL formula, formed with respect to an alternating epistemic transition system $S = \langle \Pi, \Sigma, Q, \sim_1, \ldots, \sim_n, \pi, \delta \rangle$, is one of the following:

(S0) $\top$

(S1) p, where $p \in \Pi$ is a primitive proposition;

(S2) $\neg\varphi$ or $\varphi \vee \psi$, where φ and ψ are formulae of ATEL;

(S3) $\langle\langle \Gamma \rangle\rangle \bigcirc \varphi$, $\langle\langle \Gamma \rangle\rangle \Box \varphi$, or $\langle\langle \Gamma \rangle\rangle \varphi \mathcal{U} \psi$, where $\Gamma \subseteq \Sigma$ is a set of agents, and φ and ψ are formulae of ATEL;

(S4) $K_a\varphi$, where $a \in \Sigma$ is an agent, and φ is a formula of ATEL;

(S5) $C_\Gamma\varphi$ or $E_\Gamma\varphi$, where $\Gamma \subseteq \Sigma$ is a set of agents, and φ is a formula of ATEL.

We interpret formulae of ATEL with respect to AETS, as introduced in the preceding section. Formally, if S is an AETS, q is a state in S, and φ is a formula of AETL over S, then we write $S, q \models \varphi$ to mean that φ is satisfied (equivalently, true) at state q in system S. The rules defining the satisfaction relation $\models$ are as follows:

- $S, q \models \top$
- $S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Pi$);
- $S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

- $S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;
- $S, q \models \langle\langle\Gamma\rangle\rangle \bigcirc \varphi$ iff there exists a set of strategies F_Γ, one for each $a \in \Gamma$, such that for all $\lambda \in out(q, F_\Gamma)$, we have $S, \lambda[1] \models \varphi$;
- $S, q \models \langle\langle\Gamma\rangle\rangle \Box \varphi$ iff there exists a set of strategies F_Γ, one for each $a \in \Gamma$, such that for all $\lambda \in out(q, F_\Gamma)$, we have $S, \lambda[u] \models \varphi$ for all $u \in I\!N$;
- $S, q \models \langle\langle\Gamma\rangle\rangle \varphi \,\mathcal{U}\, \psi$ iff there exists a set of strategies F_Γ, one for each $a \in \Gamma$, such that for all $\lambda \in out(q, F_\Gamma)$, there exists some $u \in I\!N$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$;
- $S, q \models K_a\varphi$ iff for all q' such that $q \sim_a q'$: $S, q' \models \varphi$;
- $S, q \models E_\Gamma\varphi$ iff for all q' such that $q \sim^E_\Gamma q'$: $S, q' \models \varphi$;
- $S, q \models C_\Gamma\varphi$ iff for all q' such that $q \sim^C_\Gamma q'$: $S, q' \models \varphi$.

Before proceeding, we introduce some derived connectives: these include the remaining connectives of classical propositional logic ($\bot \hat{=} \neg\top$, $\varphi \to \psi \hat{=} \neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi \hat{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$), together with some other useful connectives of temporal logic.

$$\begin{array}{rcl} \langle\langle\Gamma\rangle\rangle \Diamond \varphi & \hat{=} & \langle\langle\Gamma\rangle\rangle \top \,\mathcal{U}\, \varphi \\ \langle\langle\Gamma\rangle\rangle \Box^+ \varphi & \hat{=} & \langle\langle\Gamma\rangle\rangle \bigcirc \langle\langle\Gamma\rangle\rangle \Box \varphi \\ \langle\langle\Gamma\rangle\rangle \Diamond^+ \varphi & \hat{=} & \langle\langle\Gamma\rangle\rangle \bigcirc \langle\langle\Gamma\rangle\rangle \Diamond \varphi \end{array}$$

As well as asserting that some collection of agents is able to bring about some state of affairs, we can use the dual "$[\![\ldots]\!]$" to express the fact that a group of agents cannot *avoid* some state of affairs. Thus $[\![\Gamma]\!] \Diamond \varphi$ expresses the fact that the group Γ cannot cooperate to ensure that φ will never be true; the remaining agents in the system have a collection of strategies such that, if they follow them, φ will be eventually achieved. Formally, $[\![\Gamma]\!] \bigcirc \varphi$ is defined as an abbreviation for $\neg\langle\langle\Gamma\rangle\rangle \bigcirc \neg\varphi$, while $[\![\Gamma]\!] \Diamond \varphi$ is defined as an abbreviation for $\neg\langle\langle\Gamma\rangle\rangle \Diamond \neg\varphi$, and so on. Finally, we generally omit set brackets inside cooperation modalities, (i.e., writing $\langle\langle a, b\rangle\rangle$ instead of $\langle\langle \{a, b\}\rangle\rangle$), and we will usually write $\langle\langle\rangle\rangle$ rather than $\langle\langle\emptyset\rangle\rangle$.

Applications of ATEL

We hope it is clear that ATEL is a succinct and expressive language for expressing complex properties of multiagent systems. Although this paper is not primarily concerned with the applications of ATEL, we will nevertheless attempt here to give a flavour of the kind of properties that may be expressed using it.

Since ATEL is a suitable language to cope with the *dynamics of epistemics*, it is also appropriate to analyse communication issues. First, consider a system containing a sender S, a receiver R, and an environment *env* through which messages are sent. Under certain fairness conditions (the environment does not get rid of messages forever), we can express the fact that the environment cannot prevent the sender from sending a message until it is received.

$$[\![env]\!] send_m \,\mathcal{U}\, K_R m \tag{1}$$

Many speech acts can be modeled in ATEL. A *yes/no question* by i in a state q about φ simply corresponds to the constraint that i opens two alternatives $\{q^+\}$ and $\{q^-\}$, which are similar to q, except in q^+, the agent knows φ, in the other he knows $\neg\varphi$ (cf. [13]). An *answer* is then provided by an agent that selects the appropriate choice. Moreover, it is easy to add *distributed* knowledge with operator D_Γ of Γ to ATEL. Then (2) expresses the cooperative property that the group Γ can guarantee that their implicit knowledge eventually becomes explicly known by everyone:

$$D_\Gamma\varphi \to \langle\langle\Gamma\rangle\rangle \Diamond E_\Gamma\varphi \tag{2}$$

Also, ignorance may be important, both as a pre- and as a post-condition. In security protocols, where agents i and j share some common secret (a key for instance), what you typically want is (3), expressing that i can send private information to j, without revealing the message to another agent h:

$$K_i\varphi \wedge \neg K_j\varphi \wedge \neg K_h\varphi \wedge \langle\langle i, j\rangle\rangle \Diamond (K_i\varphi \wedge K_j\varphi \wedge \neg K_h\varphi) \tag{3}$$

Common knowledge C_Γ of a group Γ is also important. In particular, one is interested in conditions that ensure that

$$C_\Gamma \langle\langle\Gamma\rangle\rangle T\varphi \qquad (T \text{ a temporal operator}) \tag{4}$$

Schema (4) expresses that it is common knowledge in the group Γ that it can bring about (next, or sometime, or always, φ). It is not clear at forehand that we have a negative result about obtaining common knowledge, since it seems we can model actions stronger than communication. For instance, we may have knowledge-producing actions, and also common-knowledge producing actions, like making an announcement. If a can make an announcement p, he can choose a set of worlds in which the transitive closure of all the accessibility relations only leads to p-worlds.

As a simple example, suppose agent 1 knows whether p, i.e., $K_1p \vee K_1\neg p$, and this is common knowledge; it is also common knowledge that 1 always tells the truth. Now, given that 1 knows p, we can model that 1 can tell the truth only to 2, or to 2 and 3 separately or he can announce p in public:

$$\langle\langle 1\rangle\rangle [\bigcirc (K_2p \wedge \neg K_3p) \wedge \bigcirc (K_2p \wedge K_3p \wedge \neg C_{\{2,3\}}p) \wedge \bigcirc (C_{\{2,3\}}p)]$$

In [6], *Knowledge Games* are investigated as a particular way of learning in multiagent systems. Epistemic updates are interpreted in a simple card game, where the aim of players is to find out the deal d of cards. Having a winning strategy then easily translates into

$$d \to \langle\langle i\rangle\rangle \Diamond (K_id \wedge \bigwedge_{i \neq j} \neg K_jd) \tag{5}$$

The applicability of ATEL goes beyond epistemic updates (where epistemic post-conditions are the rule): knowledge also plays an important role in pre-conditions, expressing knowledge-dependent abilities, as in $(K_i\varphi_i \wedge K_j\varphi_j) \to \langle\langle i, j\rangle\rangle \Diamond \psi$. A model-checking algorithm for such a property might then generate what [10] calls a *knowledge based program*. An example of such an ability is (6), expressing that if Bob knows that the combination of the safe is s, then he is able to open it (o), as long as the combination remains unchanged.

$$K_b(c = s) \to \langle\langle b\rangle\rangle (\langle\langle b\rangle\rangle \bigcirc o) \,\mathcal{U}\, \neg(c = s) \tag{6}$$

Of course, when agents make strategic choices, both epistemic pre- and post-conditions are at stake: a rational agent bases his choices upon his knowledge, and will typically try to maximize his own knowledge, at the same time minimize that of his competitors. Epistemic conditions are also needed in security communication protocols, where an agent needs to know a secret key ψ in order to read a message, to obtain new knowledge φ.

4. MODEL CHECKING FOR ATEL

It is well-known that the branching temporal logic CTL lacks expressive power — fairness, for example, cannot be expressed in "vanilla" CTL (see e.g., [18] for a recent discussion on the relative merits of CTL *versus* other temporal logics). What makes CTL so attractive from the point of view of formal methods is that the *model checking* problem for CTL is computationally cheap: given a CTL model M of size m and a CTL formula φ of size ℓ, the problem of checking whether or not φ is valid in M can be solved in time $O(m \times \ell)$. This has made it possible to implement efficient, industrial strength formal verification tools for checking whether finite state systems satisfy a CTL specification [5]. The attractive computational properties of CTL are known to carry across to the alternating temporal logic of Alur et al [3]. The fact that model checking for alternating temporal logic is tractable is particularly intriguing because this problem generalises several other interesting problems of interest. For example, the *realizability* problem — showing that it is possible to implement a system *sys* that satisfies a particular specification φ [16] — simply involves model checking the formula $\langle\langle sys\rangle\rangle\varphi$ in a "maximal" model, which encodes all possible input/output relations of the environment in which the system *sys* is to operate.

In this section, we show that the tractability of model checking for alternating temporal logic carries over to ATEL: we present a (deterministic) symbolic model algorithm for ATEL that runs in time polynomial in the size of the formula and the size of the system begin checked. The core of this algorithm is given in Figure 1: the function *eval*$(\ldots)$ takes as input a formula φ of ATEL and an alternating epistemic transition system S, and returns as output the set of states in S in which the formula is satisfied. The *eval*$(\ldots)$ function is recursive, and makes use of several subsidiary definitions:

- The function $pre : 2^{\Sigma} \times 2^{Q} \to 2^{Q}$, which takes as input a set of agents Γ and a set of states Q_1 and returns as output the set of all states Q_2 such that when the system is in one of the states in Q_2, the agents Γ can cooperate and force the next state to be one of Q_1.
- The function $img : Q \times 2^{Q \times Q} \to 2^{Q}$, which takes as input a state q and a binary relation R on Q, and returns the set of states accessible from q via R. That is, $img(q, R) = \{q' \mid (q, q') \in R\}$.

Notice that for any given inputs, both of these functions may be easily computed in time polynomial in the size of the inputs and the structure against which they are being computed; the *pre* function can be immediately derived from the δ function.

We can prove the following two key properties:

LEMMA 1. *The algorithm is correct.*

PROOF. *We need to show that if, when passed formula φ and structure $S = \langle \Pi, \Sigma, Q, \sim_1, \ldots, \sim_n, \pi, \delta\rangle$), the algorithm returns some set of states Q', then*

$$Q' = \{q \mid S, q \models \varphi\}.$$

Ignoring the obvious cases, consider where the input formula is of the form $\langle\langle\Gamma\rangle\rangle\Box\psi$. Here, the algorithm should return the set of states from which Γ can cooperate to ensure that ψ is always true. Now $\langle\langle\Gamma\rangle\rangle\Box\psi$ has a fixpoint character: an axiom of ATEL *is*

$$\langle\langle\Gamma\rangle\rangle\Box\psi \leftrightarrow \psi \wedge \langle\langle\Gamma\rangle\rangle\bigcirc\langle\langle\Gamma\rangle\rangle\Box\psi$$

It follows that $\langle\langle\Gamma\rangle\rangle\Box\psi$ can be understood as a maximal solution to the fixpoint equation

$$f(x) = \psi \wedge \langle\langle\Gamma\rangle\rangle\bigcirc x.$$

where the function f maps formulae of ATEL *to formulae of* ATEL*. The loop in lines (10)–(17) of figure 1 is a (relatively standard) algorithm for computing greatest fixpoints [5, p.63]. Similarly, $\langle\langle\Gamma\rangle\rangle\varphi\,\mathcal{U}\,\psi$ is a least fixpoint — we have the following as an axiom of* ATEL

$$\langle\langle\Gamma\rangle\rangle\varphi\,\mathcal{U}\,\psi \leftrightarrow \psi \vee (\varphi \wedge \langle\langle\Gamma\rangle\rangle\bigcirc(\langle\langle\Gamma\rangle\rangle\varphi\,\mathcal{U}\,\psi))$$

The algorithm in lines (18)–(26) is a conventional algorithm for computing least fixpoints [5, p.62].

The cases where $\varphi = K_a\psi$ and $\varphi = C_\Gamma\psi$ simply involve the computation of the img function at most $|Q|$ times, as described above. □

LEMMA 2. *The algorithm terminates.*

PROOF. *Simply note that it is recursive and analytic, in that recursive calls are only made on sub-formulae of the original input formula, with primitive propositions as the recursive base.* □

Finally, we can show:

PROPOSITION 1. ATEL *model checking is* PTIME*-complete.*

PROOF. PTIME*-hardness follows from the fact that* ATEL *subsumes* ATL*, for which the model checking problem is known to be* PTIME*-complete [3]. Membership of* PTIME *is by examination of the model checking algorithm for* ATEL*. First note that the fixpoint computations require time linear in the size of the set of states [5, pp.62–63]. Computation of common knowledge (the worst case for knowledge operators) requires the computation of a transitive closure, which can be done in time $O(n^3)$ using, e.g., Warshall's algorithm. Since each recursive call is analytic, (i.e., on a subformula of the input formula), it should be clear that the overall computation requires polynomial time in the size of the inputs.* □

5. MULTIAGENT PLANNING

An important recent development in the AI planning community is the idea of *planning as model checking* [11]. The idea is as follows. A classical planning problem can be viewed as a tuple consisting of a domain D together with a goal g to be achieved. The domain D encodes the state transition properties of the environment in which the goal must be achieved. Intuitively, a domain corresponds to one of our AETS: it encodes the actions that can be performed, who can perform them, and how the performance of these actions can change state. A goal can simply be viewed as a formula that denotes a set of states — those in which it true. Giunchiglia and colleagues had the insight that a domain D for which a plan is required can be encoded as a model M_D for the branching temporal logic CTL, and the goal g of the plan may be encoded as a state formula φ_g of CTL. Checking whether there exists a plan to achieve goal g in domain D can then be treated as a model checking problem: check whether $M_D \models \mathsf{E}\Diamond\varphi_g$ is true; the formula $\mathsf{E}\Diamond\varphi_g$ asserts that there is some path through the model which eventually leads to a state where φ_g is true. If $M_D \models \mathsf{E}\Diamond\varphi_g$, then the path that is witness to the truth of the formula in the model encodes the plan that achieves the goal. Efficient contemporary CTL model checkers can then be leveraged to build planning systems.

```
function eval(φ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩) returns a subset of Q
  if φ ∈ Π then
    return {q | q ∈ π(φ)}
  elsif φ = ¬ψ then
    return Q \ eval(ψ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
  elsif φ = ψ_1 ∨ ψ_2 then
    return eval(ψ_1, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩) ∪ eval(ψ_2, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
  elsif φ = ⟨⟨Γ⟩⟩ ◯ ψ then
    return pre(Γ, eval(ψ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩))
  elsif φ = ⟨⟨Γ⟩⟩ □ψ then
    Q_1 := Q
    Q_2 := Q_3 := eval(ψ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
    while Q_1 ⊈ Q_2 do
      Q_1 := Q_1 ∩ Q_2
      Q_2 := pre(Γ, Q_1) ∩ Q_3
    end-while
    return Q_1
  elsif φ = ⟨⟨Γ⟩⟩ψ_1 U ψ_2 then
    Q_1 := ∅
    Q_2 := eval(ψ_2, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
    Q_3 := eval(ψ_1, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
    while Q_2 ⊈ Q_1 do
      Q_1 := Q_1 ∪ Q_2
      Q_2 := pre(Γ, Q_1) ∩ Q_3
    end-while
    return Q_1
  elsif φ = K_a ψ then
    Q_1 := eval(ψ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
    return {q | img(q, ∼_a) ⊆ Q_1}
  elsif φ = E_Γ ψ then
    Q_1 := eval(ψ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
    return {q | img(q, ∼^E_Γ) ⊆ Q_1}
  elsif φ = C_Γ ψ then
    Q_1 := eval(ψ, ⟨Π, Σ, Q, ∼_1, ..., ∼_n, π, δ⟩)
    return {q | img(q, ∼^C_Γ) ⊆ Q_1}
  end-if
end-function
```

Figure 1: A model checking algorithm for ATEL.

The tractability of ATEL model checking suggests that it too can be exploited to form a potentially powerful approach to multiagent planning. The idea is somewhat similar to the CTL planning approach. Suppose we wish to develop an agent a that can be guaranteed to achieve goal φ in an environment *env*. We proceed to encode the possible interactions of the agent and environment together with the initial knowledge assumptions encoded in the accessibility relations in an AETS S_{env} with initial state q_0, and check the following:

$$S_{env}, q_0 \models \langle\langle a \rangle\rangle \Diamond \varphi$$

If the answer to this problem is positive, then the witness to its truth will be a *strategy* for a that can be guaranteed to eventually achieve φ. Strategies in ATL and ATEL, as defined in section 2, above, are essentially strong plans. They define what an agent should do in any given circumstance, given any combination of events to date.

An obvious advantage of ATEL (and indeed ATL) over CTL planning is that the approach easily extends to multiagent planning: to generate a joint plan for a group Γ to achieve φ in S from q_0 we simply check whether or not

$$S, q_0 \models \langle\langle \Gamma \rangle\rangle \Diamond \varphi.$$

If the answer to this questions is positive, then there will be *a collection of strategies*, one for each member of Γ, such that by jointly following these strategies, Γ can ensure that φ is eventually true.

The additional expressive power of ATEL over ATL means that planning problems involving *epistemic goals* can easily be expressed. For example

$$\langle\langle \Gamma \rangle\rangle \Diamond K_a \varphi$$

expresses the requirement that Γ can cooperate to ensure that a knows φ. Finally, we can also express the fact that a group Γ can cooperate to make it common knowledge in Γ' that φ.

$$\langle\langle \Gamma \rangle\rangle \Diamond C_{\Gamma'} \varphi$$

We have thus shown how multiagent planning for epistemic goals can be reduced to a model checking problem in ATEL. We know from the discussion above that model checking in ATEL is PTIME-complete, and hence tractable. Using techniques developed for symbolic model checking (e.g., binary decision diagrams), such algorithms can be efficiently implemented, even with extremely large state spaces [5].

6. CASE STUDY AND EXPERIMENTS

We now present a short case study, illustrating our ideas. We also discuss some experiments we have conducted using the MOCHA model checking system to evaluate our ideas.

The Train Controller

The system we consider is a train controller (adapted from [1]). The system contains three agents: two trains, and a controller — see Figure 2(a). The trains, one of which is Eastbound, the other of which is Westbound, each occupy their own circular track. At one point, both tracks pass through a narrow tunnel — there is not room for both trains in the tunnel at the same time. There are traffic lights on both sides of the tunnel, which can be either red or green. Both trains are equipped with a signaller, with which they can send signals to the controller; the idea is that they send a signal when they approach the tunnel. The controller can receive signals from both trains, and controls the color of the traffic lights. The task of the controller is, first and foremost, to ensure that the trains are never both in the tunnel at the same time; the secondary task is to ensure the "smooth running" of the system (e.g., the trains can always move through the tunnel, they cannot be forced into the tunnel, and so on).

The MOCHA System

The train controller system was modelled by Alur and colleagues using a prototype model checking system for ATL called MOCHA [2, 1]. MOCHA takes as input an alternating transition system described using a (relatively) high level language called REACTIVE-MODULES, which loosely resembles high level programming languages such as C. The system is then capable of either randomly simulating the execution of this system, or else of taking formulae of ATL and automatically checking their truth or falsity in the transition system. As well as ATL formulae, MOCHA is capable of *invariant* checking — of checking whether a given property is true across all reachable states of the system. Although in its current implementation, MOCHA is capable of model checking arbitrary ATL formulae, (and hence of determining whether or not there exists a collective strategy to achieve some multiagent goal), it does not exhibit such a strategy; it merely announces whether one exists.

In the model developed by Alur and colleagues, each train was modelled by an automaton that could be in one of three states (see Figure 2(b)): "away" (state s_0 — the initial state of the train); "wait" (state s_1 — waiting for a green light to enter the tunnel); and "tunnel" (state s_1 — the train is in the tunnel). Transitions between states may be guarded: for example, in order for a train to go from s_1 to s_2, the condition "signal is green" must be true. If a state transition is not labelled with a condition, then the condition is assumed to be always true. In addition, when an agent makes a transit from one state to another, it may send a signal, as indicated by dashed lines in Figure 2(b). So, for example, when a train is entering the tunnel, it sends a signal to the controller to this effect. Note that just because a train *can* make a state transition does not necessarily mean it does so: it may be "lazy" (in the terminology of MOCHA), staying in the same state.

The train controller itself starts by setting both traffic lights to red. When a train approaches the tunnel (indicated by a "entering the tunnel" signal), the controller checks whether the opposing light is red; if it is, then the light for the approaching train is set green, allowing access. When a train moves out of the tunnel (also indicated by a signal to the controller, the controller sets the light associated with this train to red.

Checking Epistemic Properties with MOCHA

In [1], various possible ATL properties of this system are discussed, and may be automatically checked using MOCHA. However, currently MOCHA does not support the knowledge modalities of ATEL. We now discuss a preliminary approach we have developed to check knowledge properties using MOCHA, which involves translating knowledge formulae into ATL. The idea is inspired by translation-based theorem proving methods for modal logics, which exploit the fact that formulae of modal logic can be automatically translated into first-order logic.

The main component of ATEL missing from MOCHA is the accessibility relations used to give a semantics to knowledge. Where do these relations come from? We use the *interpreted systems* approach of [10]. Given a state $q \in Q$ and agent $a \in \Sigma$, we write $state_a(q)$ to denote the *local* state of agent a when the system is in state q. The agent's local state includes its program counter and all its local variables. We then define the accessibility relation $\sim_a$ as follows:

$$q \sim_a q' \quad \text{iff } state_a(q) = state_a(q'). \tag{7}$$

We emphasise that this approach is well known and widely used in the distributed systems and epistemic logic communities. So, suppose we want to check whether, when the system is in some state q, agent a knows φ, i.e., whether $S, q \models K_a\varphi$. Then by (7), this amounts to showing that

$$\forall q' \in Q \text{ s.t. } state_a(q) = state_a(q') \text{ we have } S, q' \models \varphi. \tag{8}$$

We can represent such properties directly as formulae of ATL, which can be automatically checked using MOCHA. In order to do this, we need some additional notation:

- we express the value of $state_a(q)$ as a constant s; and
- we have a logical variable $state_a$ that denotes, in any given state q , the value of $state_a(q)$.

Then we can express (8) as the following ATL invariant formula:

$$\langle\langle\rangle\rangle \Box((state_a = s) \rightarrow \varphi) \tag{9}$$

Note that in saying that (9) is an invariant, we are stating that it must hold across all reachable states of the system. Another way of reading (9) is as "agent a's state *s carries the information that* φ".

Such formulae can be directly written as properties that can be checked using MOCHA:

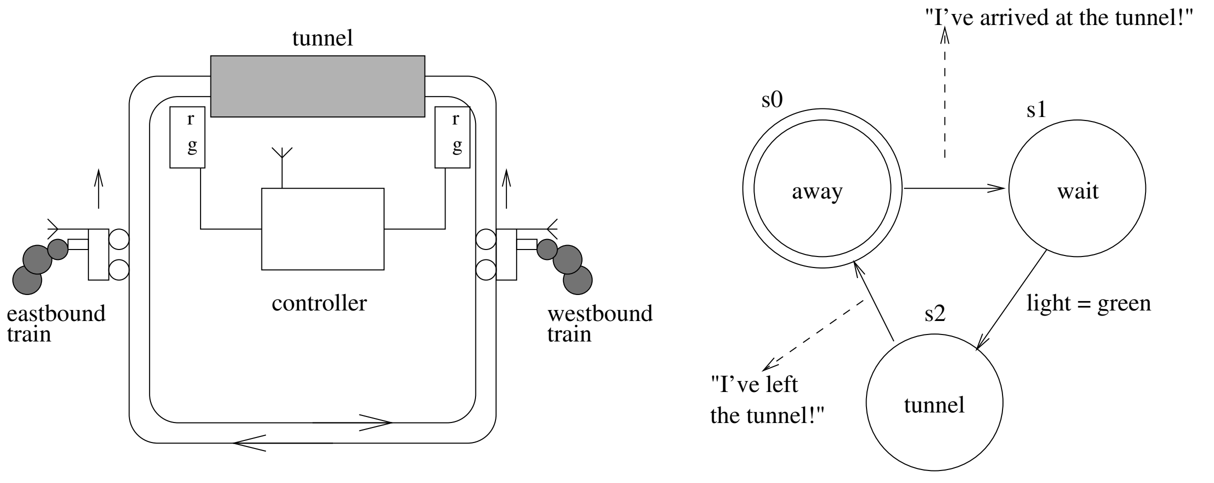
```
<< >>G((stateA = s) -> phi)
```

The G here is the MOCHA text form of the "always" operator ("$\Box$").

Turning back to the train example, we now show how a number of knowledge properties of the system were proven. First, consider the property that "when one train is in the tunnel, it knows the other train is not in the tunnel":

$$(state_a = tunnel) \rightarrow K_a(state_b \neq tunnel) \quad (a \neq b \in \{E, W\})$$

Translating into the MOCHA text form of ATL this schema gives the following two formulae



(a) Overall structure of the train controller sytem

(b) Train states, transitions, and signals

Figure 2: The train controller system.

```
<<>> G ((stateE=tunnel) => ~(stateW=tunnel))
<<>> G ((stateW=tunnel) => ~(stateE=tunnel))
```

which were successfully model checked.

We can also show that when a train is away from the tunnel, it does not know whether or not the other train is in the tunnel.

$$\langle\langle\rangle\rangle \Box (state_a \neq tunnel) \rightarrow [(\neg K_a(state_b = tunnel)) \wedge (\neg K_a(state_b \neq tunnel))] \qquad (a \neq b \in \{E, W\})$$

For the westbound train, we do this by checking the following formulae, both of which fail.

```
<<>> G ~(stateE=tunnel) => (stateW=tunnel)
<<>> G ~(stateE=tunnel) => ~(stateW=tunnel)
```

We can conclude that the *only way a train knows whether the other train is in the tunnel is if it is in the tunnel itself* (in which case it knows the other is not).

$$\langle\langle\rangle\rangle \Box [(K_a(state_b \neq tunnel)) \leftrightarrow (state_a = tunnel)]$$

(With $a \neq b \in \{E, W\}$.) We can also check properties relating knowledge and ability. For example, we can prove that an agent always knows that the system can cooperate with it to allow it eventual access.

$$\langle\langle\rangle\rangle \Box K_a \langle\langle\Sigma\rangle\rangle \Diamond (state_a = tunnel) \qquad (a \in \{E, W\})$$

Since we wish to show that an agent always knows something, the quantification over agent a's knowledge-accessible states is across *all* states of the system. We can thus write, for the westbound train (`C` is the controller, and `F` is the MOCHA form of "$\Diamond$"):

```
<<>> G <<C,TrainW,TrainE>> F (stateW=tunnel)
```

In fact, since we are quantifying over *all* states of the system, this gives us that it is *always common knowledge that the grand coalition of all agents can cooperate to eventually get train a in the tunnel*:

$$\langle\langle\rangle\rangle \Box C_\Sigma \langle\langle\Sigma\rangle\rangle \Diamond (state_a = tunnel) \qquad (a \in \{E, W\}) \tag{10}$$

Planning for Epistemic Goals

Recall, from the discussion in preceding chapters, the general structure of formulae we need to check in order to generate plans for epistemic goals: $\langle\langle\Gamma\rangle\rangle \Diamond K_a \varphi$ means that Γ can cooperate to make a know φ. Without quantification, which we do not have in MOCHA, this property is not expressed so easily — but it is possible. To see how we might do this, assume that agent a can be in n distinct states $s_1, \ldots, s_n$. Then saying that Γ can bring about knowledge of φ in agent a is the same as saying:

- agent a's state s_1 carries information φ and Γ can ensure that a enters s_1; or
- agent a's state s_2 carries information φ and Γ can ensure that a enters s_2; or…
- agent a's state s_n carries information φ and Γ can ensure that a enters s_n; or

This observation allows us to rewrite $\langle\langle\Gamma\rangle\rangle \Diamond K_a \varphi$ in ATL as:

$$\bigvee_{1 \leq i \leq n} (\langle\langle\rangle\rangle \Box ((state_a = s_i) \rightarrow \varphi) \wedge \langle\langle\Gamma\rangle\rangle \Diamond state_a = s_i)$$

Such ATL formulae can be directly coded and checked in MOCHA.

The first property we prove relates to a train's knowledge about whether or not the other train is in the tunnel. Consider: is it possible to cause a train to know that the other is not in the tunnel? We saw above that when one train is in the tunnel, it knows that the other is not; but when a train is away from the tunnel, it has no definite knowledge about the position of the other train. So, for a train to know that the other train is not in the tunnel, *it must be in the tunnel*. The first property we can check is that the grand coalition of agents can cooperate to make a train know that the other is not in the tunnel:

$$\langle\langle\rangle\rangle \Box \langle\langle\Sigma\rangle\rangle \Diamond K_a(state_b \neq tunnel) \qquad (a \neq b \in \{E, W\})$$

For the westbound train, this property when translated and slightly simplified becomes the following property, which can readily be checked in MOCHA.

```
<<>>G (<<TrainW,C,TrainE>> F (stateW=tunnel))
```

Interestingly, *no other subset of agents can bring this knowledge about* — because no other subset of agents can be guaranteed to get the westbound train into the tunnel. Thus, for example, the following property does not hold.

$$\langle\langle\rangle\rangle \Box \langle\langle a \rangle\rangle \Diamond K_a(state_b \neq tunnel) \qquad (a \neq b \in \{E, W\})$$

From (10), we know that it is always common knowledge that the entire system can cooperate to get a train in the tunnel. We can thus conclude that it is always common knowledge that the entire system can cooperate to eventually cause train a to know that train b is not in the tunnel:

$$\langle\langle\rangle\rangle \Box C_\Sigma \langle\langle \Sigma \rangle\rangle \Diamond K_a(state_b \neq tunnel) \qquad (a \neq b \in \{E, W\})$$

7. CONCLUDING REMARKS

In this paper, we have introduced a natural extension to the Alternating Temporal Logic of Alur and colleagues, which includes modalities for representing knowledge, common knowledge, and the like. Using a simple example, we illustrated how an existing model checker can be put to work to verify formulas in ATEL, and in particular to determine the existence or otherwise of collective plans for bringing about epistemic goals. The witness of such an existence proof often is a *strategy* or, for the multi-agent case, a set of strategies, or *plans*.

We think that our approach can be extended to deal with more complex examples, including both theoretical ones, (like the muddy children [10]), or in multi-agent systems in which information hiding or enclosing may be options for the agents (cf. [17], where mailmen can benefit from keeping some of their tasks secret). We do not know yet which fragment of ATEL can be easily translated into ATL, but maybe the use of *local propositions* ([9]) may substantially broaden this fragment (see also [12], where those propositions were used to translate epistemic goals into linear temporal logic).

Recently, there has been a lot of emphasis on modelling knowledge and its dynamics in one and the same framework [6, 4], which, in turn, generalises the mas-approaches to belief revision. It is clear that ATEL offers a framework to facilitate this. Moreover, many impressive platforms have emerged that integrate (the dynamics of) epistemics, rationality and decision making. Enhancement of the work begun in this paper might further the computational relevance of such integrated theories. This is especially so, since the focus of the mentioned platforms has thus far been on formalising epistemic notions in game-theoretic settings. The question of how to use these formalisations in finding winning strategies in games of imperfect information for example, has only recently been asked (cf. [7]). Finally, also the social choice community has discovered ATL [15]. We firmly believe that epistemic operators provide a valuable tool for allowing agents to reason about which coalition they need or might like to join.

8. REFERENCES

[1] R. Alur, L. de Alfaro, T. A. Henzinger, S. C. Krishnan, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. MOCHA user manual. University of Berkeley Report, 2000.

[2] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[3] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 100–109, Florida, October 1997.

[4] A. Baltag. A logic for supicious players. *Bulletin of Economic Research*, 54(1):1–45, 2002.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press: Cambridge, MA, 2000.

[6] H.P. van Ditmarsch. *Knowledge Games*. PhD thesis, University of Groningen, Groningen, 2000.

[7] S. Druiven. Opponent modeling and dynamic epistemic logic in games with imperfect information. M.Sc. thesis, in preparation, 2002.

[8] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[9] K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In *Proceedings of the 1998 Conference on Theoretical Aspects of Reasoning about Knowledge (TARK98)*, pages 29–41, Evanston, IL, July 1998.

[10] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA, 1995.

[11] F. Giunchiglia and P. Traverso. Planning as model checking. In S. Biundo and M. Fox, editors, *Recent Advances in AI Planning (LNAI Volume 1809)*, pages 1–20. Springer-Verlag: Berlin, Germany, 1999.

[12] W. van der Hoek and M.J.W. Wooldridge. Model checking knowledge and time. In D. Bošnački and S. Leue, editors, *Model Checking Software — Proceedings of SPIN 2002 (LNCS Volume 2318)*, pages 95–111. Springer-Verlag: Berlin, Germany, 2002.

[13] B. van Linder, W. van der Hoek, and J.-J. Ch. Meyer. Actions that make you change your mind. In A. Laux and H. Wansing, editors, *Knowledge and Belief in Philosophy and AI*, pages 103–146. Akademie-Verlag, 1995.

[14] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press: Cambridge, England, 1995.

[15] M. Pauly. A logical framework for coalitional effectivity in dynamic procedures. *Bulletin of Economic Research*, 53(4):305–324, 2002.

[16] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *Proceedings of the Sixteenth International Colloquium on Automata, Languages, and Programs*, 1989.

[17] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press: Cambridge, MA, 1994.

[18] M. Y. Vardi. Branching vs. linear time: Final showdown. In T. Margaria and W. Yi, editors, *Proceedings of the 2001 Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001 (LNCS Volume 2031)*, pages 1–22. Springer-Verlag: Berlin, Germany, April 2001.

[19] M. Wooldridge, C. Dixon, and M. Fisher. A tableau-based proof method for temporal logics of knowledge and belief. *Journal of Applied Non-Classical Logics*, 8(3):225–258, 1998.