Model Checking ATL and its epistemic extensions

Franco Raimondi, Alessio Lomuscio
Department of Computer Science
University College London, UK
{f.raimondi, a.lomuscio}@cs.ucl.ac.uk

March 1, 2005

Abstract

We present an OBDD-based methodology for verifying multi-agent systems specified by the logic ATL. We present an implementation, discuss traditional multi-agent systems examples, and report experimental results by comparing the implementation to MOCHA, a state-of-the-art model checker for ATL.

1 Introduction

Model checking is a formal verification technique that allows for the automatic verification of large, distributed systems. The key idea of model checking is to represent a system by means of a semantical model, and a property of the system by means of a logical formula. Model checking is the process of verifying (possibly automatically) that a system complies with a required property; this is done by verifying that a formula is true in a model.

Model checking was traditionally put forward to verify specifications given in *temporal* logics [7]. Recently, however, researchers have extended model checking techniques to other modal logics, including some typical multi-agent systems (MAS) logics, thereby allowing to verify formally a range of multi-agent systems. Examples of efforts on this line include [27, 4, 9, 21, 23, 18]. These works share the model checking approach but differ in the logic specification language, and in the specific model checking technique.

We aim to make further progress in this line by analysing the formal verification of epistemic extensions of the standard Alternating-time Temporal Logic (ATL). ATL was introduced by Alur et al [2] to reason about *strategies* in multi-player games. A model checker for this logic has been developed [1]. As noticed in [10], the logical models for ATL share many similarities with logical models for MAS and, following this intuition, van der Hoek and Wooldridge proposed the logic ATEL [11]. ATEL is an extension of ATL with epistemic operators, whose semantics is based on multi-agent systems and not on game structures. However, it has been argued [14, 17, 15, 16] that

ATEL semantics differs from the original ATL semantics. In particular, [17, 15, 16] have proposed a different semantics for ATL operators in MAS, henceforth referred to as ATOL, while [26] has clarified the original ATEL proposal. Without wishing to continue this debate, it seems to us that both logics are worth exploring, as they seem to express different properties. ATEL stresses what agents *may* bring about by guessing moves; in this sense, ATEL is more suitable to reason about unwanted states of affairs (see Section 5.2). On the other hand, ATOL seems more appropriate when reasoning about *feasible* plans (see Section 5.1). Against this background, and in parallel with this discussion, [11, 10, 22] suggested different techniques to reduce the problem of ATEL model checking to standard ATL model checking, with the idea of using MOCHA, the only existing model checker for ATL, for model checking MAS.

In this paper we consider model checking algorithms for ATEL and ATOL based on ordered binary decision diagrams (OBDD's), and we present MCMAS, a model checker for the automatic verification of ATEL and ATOL formulae The tool is available for download, together with the examples presented below, under the terms of the GPL license. MCMAS has a dedicated programming language (ISPL), that allows for the efficient specification of MAS in a natural way. Differently from previous approaches, our tool does not involve the translation or the reduction of the problem of model checking to plain ATL, and it allows for the automatic verification of ATOL formulae which, to our knowledge, is not supported by any implementation.

The rest of the paper is organised as follows. In Section 2 we review the logic ATL, some basic model checking methodologies, and the formalism of interpreted systems upon which our tool is based. In Section 3 we discuss ATEL and ATOL, and we introduce two algorithms for model checking these logics. In Section 4 we present the tool that implements these algorithms. In Section 5 we verify two examples by means of our tool, and we discuss again the properties of ATEL and ATOL. In Section 6 we compare the performance of our tool against MOCHA, and we conclude in Section 7.

2 Preliminaries

In this section we review the main formalisms that we shall use in the remainder of this paper. We first introduce the logic ATL, and then we present the main concepts of OBDD-based model checking; finally, we describe the formalism of interpreted systems to model multi-agent systems. We will consider extensions of ATL to multi-agent systems in Section 3.1, where our choices for the chosen model checking algorithm will be argued.

2.1 ATL

The syntax of the temporal logic ATL (Alternating-time Temporal Logic) [2] is defined as follows. Let AP be a finite set of atomic propositions, let $\Sigma = \{1, \ldots, n\}$ be a set of players, and let $\Gamma \subseteq N$ be a subset of the set of players. Well-formed ATL formulae are defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle\Gamma\rangle\rangle X\varphi \mid \langle\langle\Gamma\rangle\rangle G\varphi \mid \langle\langle\Gamma\rangle\rangle(\varphi U \psi)$$

In [2], the formula $\langle\langle\Gamma\rangle\rangle\varphi$ is read as "the set of agents Γ can enforce φ". ATL can thus be seen as a refinement of the temporal logic CTL [7] where the path quantifiers E and A are replaced with quantification over a set of players. Indeed, quantification over the set of all players is read as the existential quantifier of CTL, while quantification over the empty set of agent is read as the universal quantifier of CTL.

The semantics of ATL formulae is given in terms of *concurrent game structures* (CGS). A CGS is a tuple $< \Sigma, S, AP, h, d, \delta >$ where Σ is a set of players, S is a finite set of states, AP is a set of atomic propositions, $h : AP \rightarrow 2^S$ is a labelling function, $d_i : \Sigma \times S \rightarrow I\!N$ is the number of moves available to a player i in a state (moves are labelled with natural numbers), and $\delta : S \times d_1 \times \ldots \times d_N \rightarrow S$ is an evolution function that associates a state to a (current) state and a set of moves, one for each player. A *strategy for player* i is a function f_i that maps sequences of states to a natural number, corresponding to a move available to player i at the end of the sequence: $f_i : S^+ \rightarrow I\!N$, such that $f_i(s) < d_i(s)$ for all states in the sequence. Given a state $s \in S$, a set of players Γ, and a set of strategies $F_\Gamma = \{f_i | i \in \Gamma\}$, the set $out(s, F_\Gamma) \subseteq S^+$ is the set of sequences that the group Γ can *enforce* in s. A sequence of states $s_0, s_1, \ldots$ is denoted with π, and $\pi(i) = s_i$ denotes the state at place i in the sequence. Satisfaction of an ATL formula in a state $s \in S$ of a given CGS is defined as follows:

$s \models p$	iff	$s \in h(p)$,
$s \models \neg\varphi$	iff	$s \not\models \varphi$,
$s \models \varphi_1 \vee \varphi_2$	iff	$s \models \varphi_1$ or $s \models \varphi_2$,
$s \models \langle\langle\Gamma\rangle\rangle X\varphi$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(s, F_\Gamma)$, $\pi(1) \models \varphi$.
$s \models \langle\langle\Gamma\rangle\rangle G\varphi$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(s, F_\Gamma)$ and $\forall i \geq 0$, $\pi(i) \models \varphi$.
$s \models \langle\langle\Gamma\rangle\rangle(\varphi U \psi)$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(s, F_\Gamma)$, $\exists i \geq 0$ s.t. $\pi(i) \models \psi$ and $\forall 0 \leq j < i$, $\pi(j) \models \varphi$

It is worth noticing here that the definitions above assume that *every player has complete information about the system*. Under this assumption, it has been proven in [2] that the model checking problem for ATL is PTIME-complete. If a player has *incomplete information*, then his strategy can depend only on the *observable* part of the history of the game; in this case, the model checking problem for ATL is undecidable (see [2], p.708).

2.2 OBDD-based model checking

The problem of **model checking** can be defined as establishing whether or not a model M satisfies a formula φ ($M \models \varphi$). Though M could be a model for any logic, traditionally the problem of building tools to perform model checking automatically has been investigated for *temporal* logics [7, 13], and more recently for ATL [1]. The model M is usually represented by means of a dedicated programming language, such as PROMELA[12], SMV [20], and REACTIVEMODULES [1]. In many approaches, the model for the program is not built explicitly, but *symbolically*. Techniques to achieve this are based on ordered binary decision diagrams, SAT translations [3], or other algebraic structures. These approaches are often referred to as *symbolic model checking* techniques; other approaches exist, notably with automata [12]. For the purposes of this paper, we review briefly symbolic model checking using OBDD's.

OBDD's are an efficient representation for the manipulation of boolean functions. OBDD's of different functions can be composed efficiently: in [5] algorithms are provided for the manipulation and the composition of OBDD's.

The key idea of model checking a logic by using OBDD's is to represent states and relations in a model by means of boolean formulae. Any formula of the logic is then identified with a set of states, i.e. the states of the model satisfying the formula. As set of states can be represented as a boolean formula, each formula of the logic can be characterised by a boolean formula. Thus, the problem of model checking can be reduced to the construction of boolean formulae. This is achieved by composing OBDD's, or by computing fix-points of operators on OBDD's; we refer to [2, 13] for more details. Using this technique, systems with a state space in the region of 10^{40} have been verified for the temporal logic CTL.

2.3 Interpreted systems

In this section we introduce the formalism of interpreted systems. We will sometimes overload the notation used in Section 2.1, to underline the similarities between CGS and interpreted systems.

An **interpreted system** [8] is a semantic structure representing a set of agents $\Sigma = \{1, \ldots, n\}$. Each agent $i \in \Sigma$ is characterised by a finite set of *local states* L_i and by a finite set of actions Act_i that may be performed. Actions are performed in compliance with a protocol $P_i : L_i \rightarrow 2^{Act_i}$. Notice that this definition of protocols allows for *non-determinism* in the system. The environment in which agents "live" may be modelled by means of a special agent E, modelled by a set of local states L_E, a set of actions Act_E, and a protocol P_E. A tuple $g = (l_1, \ldots, l_n, l_e) \in L_1 \times \ldots \times L_n \times L_E$, where $l_i \in L_i$ for each i, is called a *global state* and gives a description of the system at a particular instance of time. The evolution of the agents' local states is described by a function $t_i : L_i \times L_E \times Act_1 \times \ldots \times Act_n \rightarrow L_i$ which gives the "next" local state as a function of the current local state of the agent, the environment, and all the other agents' actions. We assume that, in every state, agents evolve simultaneously (such a system is usually referred to as *lock-step* system). The evolution of the (global states of the) system may be described by a function $t : G \times Act \rightarrow G$, where $G \subseteq (L_1 \times \ldots \times L_n \times L_E)$ denotes the set of *reachable* global states, and $Act = Act_1 \times \ldots \times Act_n \times Act_E$ denotes the set of joint actions. The function t is the composition of all the functions t_i, and it is defined by $t(g, a) = g'$ iff $\forall i, t_i(l_i(g), a) = l_i(g')$, where $l_i(g)$ denotes the local state of agent i in global state g. The set G of reachable global states is obtained by considering all the possible evolutions of the system from a set of *initial* global states, denoted with I. Finally, to complete the description of a MAS, a set of atomic propositions AP is introduced, together with a valuation function $h : AP \rightarrow 2^G$.

In symbols, we will denote an interpreted systems IS with a tuple $IS = \langle (L_i, Act_i, P_i, t_i)_{i \in \Sigma}, I, h \rangle$. Interpreted systems have been proven a suitable semantics for reasoning about temporal and epistemic properties of agents [8, 19]. Model checking algorithms and tools have been developed for the verification of epistemic and temporal properties of MAS expressed in the formalism of interpreted systems [9, 23].

3 Model checking ATL and its epistemic extensions

Some problems arise when evaluating ATL formulae on semantical models designed for multi-agent systems. We review the relevant literature on this subject in Section 3.1. In Section 3.2 we introduce two different algorithms for the verification of ATL in interpreted systems systems using OBDD's.

3.1 ATL and multi-agent systems

In Section 2.1 and 2.3 the same symbol Σ was used to denote a set of players and a set of agents. Indeed, many similarities are evident between CGS's and interpreted systems. In both formalisms there are individual actors (the players and the agents), and these actors are allowed to perform certain moves (or actions), depending on the state in which they are. Moves (or actions) label transitions between states; these transitions generate a temporal evolution of the system, starting from a set of initial states. In this line, it seems natural to reason about strategies for agents in MAS, and not only for players of a game.

3.1.1 ATEL

In [11], the logic ATEL, an extension of ATL, is introduced. In addition to cooperation modalities, ATEL includes epistemic operators K_i, one for each agent $i \in \Sigma$, and operators for group modalities to characterise common knowledge, distributed knowledge, and knowledge in a group of agents. The semantics of ATEL formulae may be given in terms of an interpreted system IS ([11] uses *epistemic transition systems*, but the differences are minimal). We define a *strategy for agent* i as a function from global states to a set of actions: $f_i : G \to 2^{Act}$ (notice that we are dropping here the assumption of *perfect recall* as assumed in [2]). As in Section 2.1, F_Γ denotes a set of strategies for the group of agents Γ, and $out(g, F_\Gamma)$ denotes the set of sequences (or computations) from global state g that the group Γ can enforce. Satisfaction of an ATEL formula φ in global state g of an interpreted system IS is defined as follows:

$g \models p$	iff	$g \in h(p)$,
$g \models \neg\varphi$	iff	$g \not\models \varphi$,
$g \models \varphi_1 \vee \varphi_2$	iff	$g \models \varphi_1$ or $g \models \varphi_2$,
$g \models \langle\langle\Gamma\rangle\rangle X\varphi$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(g, F_\Gamma)$, $\pi(1) \models \varphi$.
$g \models \langle\langle\Gamma\rangle\rangle G\varphi$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(g, F_\Gamma)$ and $\forall i \geq 0, \pi(i) \models \varphi$.
$g \models \langle\langle\Gamma\rangle\rangle(\varphi U \psi)$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(g, F_\Gamma), \exists i \geq 0$ s.t. $\pi(i) \models \psi$ and $\forall 0 \leq j < i, \pi(j) \models \varphi$
$g \models K_i(\varphi)$	iff	for all g' s.t. $l(g) = l(g')$, $g' \models \varphi$.

(we refer to [11, 8] for the remaining group modalities). We say that a formula φ is true in an interpreted system IS, written $IS \models \varphi$, if $g \models \varphi, \forall g \in I$. The authors of [11] suggest that the problem of model checking ATEL formulae can be reduced to verification of ATL formulae, using the MOCHA model checker [1], and they present an example of this translation.

The interpretation of ATL formulae in multi-agent system models, however, raises some issues on the interpretation of the ATL operators. These problems have been

noted by van der Hoek and Wooldridge in [26], and by other authors [14, 17, 15, 16]. Indeed, unlike in CGS, in the formalism of interpreted systems:

1. agents have *incomplete* information about the system (in the sense of [2]), because agents are not allowed to observe the other agents' local states (therefore, they cannot distinguish between *epistemically equivalent* global states);

2. the protocols are non-deterministic, and agents may perform different actions in the same state, or in two epistemically equivalent states.

These differences may cause counter-intuitive properties to hold true in certain models. This happens, for example, in the simple card game introduced in [14, 17] and discussed in Section 5.1. It has been suggested by van der Hoek and Wooldridge [26] that the ATEL operator $\langle\langle\Gamma\rangle\rangle$ *does not* express the idea that the group Γ has a a strategy to *enforce* a state of affairs. Instead, $\langle\langle\Gamma\rangle\rangle$ may be read as *group* Γ has the possibility of bringing about something, perhaps by "guessing" the moves in epistemically equivalent states. Although this is not the intended meaning of the operators as given in [2], it seems to us that this interpretation is useful in certain circumstances; we provide a motivational example of this in Section 5.2.

3.1.2 ATOL

If, differently from above, one wants to remain close to the original interpretation of $\langle\langle\Gamma\rangle\rangle$ as being able to *enforce* a state of affairs, changes are needed in the interpretation of ATL operators over interpreted systems. The key idea, presented in [16, 17], is to restrict the evaluation of the ATL operators to *uniform* (or *feasible*) strategies in an interpreted system. A strategy is said to be *uniform* if (1) it is a valid strategy, in the sense of Section 3.1.1, and (2) if two global states are related via an epistemic relation for agent i, then agent i must perform the same action in the two states. Instead of considering a single agent, in point (2) above it is possible to consider a *collective* strategy for a group of agents and impose various conditions of the accessibility relations. We refer to [16] for more details. For the purposes of this paper, when considering a group of agents, we will impose that each agent in the group performs the same action in epistemically equivalent states.

For the purposes of this paper, we are not interested in the full syntax of ATOL (as presented in [16]), and we do not see major issues in the extension of what is presented below to the full ATOL. We will consider the following subset of the language of ATOL:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle\Gamma\rangle\rangle^{\bullet} X\varphi \mid \langle\langle\Gamma\rangle\rangle^{\bullet} G\varphi \mid \langle\langle\Gamma\rangle\rangle^{\bullet}(\varphi U \psi) | K_i(\varphi)$$

In [16, 17], the semantics of this language is given as follows:

$g \models p$	iff	$g \in h(p)$,
$g \models \neg\varphi$	iff	$g \not\models \varphi$,
$g \models \varphi_1 \vee \varphi_2$	iff	$g \models \varphi_1$ or $g \models \varphi_2$,
$g \models \langle\langle\Gamma\rangle\rangle^{\bullet} X\varphi$	iff	there exists a set of *uniform strategies* F_Γ s.t. for all computations $\pi \in out(g, F_\Gamma)$, $\pi(1) \models \varphi$.
$g \models \langle\langle\Gamma\rangle\rangle^{\bullet} G\varphi$	iff	there exists a set of *uniform strategies* F_Γ s.t. for all computations $\pi \in out(g, F_\Gamma)$ and $\forall i \geq 0$, $\pi(i) \models \varphi$.
$g \models \langle\langle\Gamma\rangle\rangle^{\bullet} (\varphi U \psi)$	iff	there exists a set of *uniform strategies* F_Γ s.t. for all computations $\pi \in out(g, F_\Gamma)$, $\exists i \geq 0$ s.t. $\pi(i) \models \psi$ and $\forall 0 \leq j < i$, $\pi(j) \models \varphi$
$g \models K_i(\varphi)$	iff	for all g' *compatible with the strategy* and s.t. $l(g) = l(g')$, $g' \models \varphi$.

To interpret ATOL formulae in interpreted systems IS, we proceed as follows. We consider the set Θ of all the uniform strategies for the agents in IS. For each strategy $s \in \Theta$, we build a new interpreted system $IS_s (s \in \Theta)$, which is identical to IS, except that the protocols for the agents must be consistent with the uniform strategy s (i.e. the protocols must prescribe the same action in epistemically equivalent states). In line with the original ATOL semantics, we evaluate an ATOL formula as true in IS if the formula holds in at least one of the generated interpreted systems $\{IS_s\}_{s\in\Theta}$ We provide more details about the interpretation of ATOL formulae in Section 3.2.2.

3.2 Model checking algorithms

In this section we present two algorithms for the verification of ATEL and ATOL formulae in an interpreted system. Our approach is similar, in spirit, to the traditional model checking techniques for the temporal logic CTL [7], and to the approach presented in [23]. Indeed, we reduce the problem of verification of a formula to the verification of the equivalence of two boolean formulae. In particular, we proceed as follows. Given an interpreted system $IS = \langle (L_i, Act_i, P_i, t_i)_{i\in\Sigma}, I, h\rangle$, we begin by computing the number of boolean variables $v_i (i \in I\!N)$ required to encode the local states of an agent, denoted with $nv(i)$: $nv(i) = \lceil log_2 |L_i| \rceil$. Similarly, to encode an agent's action, the number of boolean variables $w_i (i \in I\!N)$ required is $na(i) = \lceil log_2 |Act_i| \rceil$. Thus, a global state g can be encoded as a boolean vector $(v_1, \ldots, v_N)$, where $N = \sum_i nv(i)$. A joint action a can be encoded as a boolean vector $(w_1, \ldots, w_M)$, where $M = \sum_i na(i)$. A set of global states (or joint actions) can be expressed as the disjunction of the boolean formulae encoding each global state in the set. Having encoded local states, global states, and actions by means of boolean variables, all the remaining parameters can be expressed as boolean functions of these variables. Indeed, since the protocols relate local states to set of actions, they can also be expressed as boolean formulae. The evolution functions can be translated into boolean formulae, too. The set of initial states is easily translated, while h can be translated into a function returning a set of states, i.e. a boolean function.

3.2.1 Model checking ATEL

In addition to the parameters presented above, the algorithm for model checking ATEL requires the definition of a boolean function $R_t(g, g')$, representing a possible transition between the global states g and g'. $R_t(g, g')$ can be obtained from the evolution

$SAT(\varphi)$ {
 φ is an atomic formula: return $h(\varphi)$;
 φ is $\neg\varphi_1$: return $G \setminus SAT(\varphi_1)$;
 φ is $\varphi_1 \wedge \varphi_2$: return $SAT(\varphi_1) \cap SAT(\varphi_2)$;
 φ is $\langle\langle\Gamma\rangle\rangle X\varphi_1$: return $SAT_X(\varphi_1, \Gamma)$;
 φ is $\langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$: return $SAT_U(\varphi_1, \varphi_2, \Gamma)$;
 φ is $\langle\langle\Gamma\rangle\rangle G\varphi_1$: return $SAT_G(\varphi_1, \Gamma)$;
 φ is $K_i\varphi_1$: return $SAT_K(\varphi_1, i)$;
}

Figure 1: ATEL algorithm

functions t_i by quantifying over actions. This quantification can be translated into a propositional formula using a disjunction (see [7] for a similar approach to boolean quantification):

$$R_t(g, g') = \bigvee_{a \in Act} [(t(g, a, g') \wedge P(g, a)]$$

where $P(g, a)$ is a boolean formula imposing that each component of the joint action a is consistent with the agents' protocols in global state g. The above gives the desired boolean relation between global states. Also, to evaluate the epistemic operator K, it is necessary to restrict the epistemic accessibility relations to *reachable* states. The set G of reachable global states can be expressed symbolically by a boolean formula, and it can be computed as the fix-point of the operator $\tau(Q) = (I(g) \vee \exists g'(R_t(g', g) \wedge Q(g'))$. The fix-point of τ can be computed by iterating $\tau(\emptyset)$ by standard procedure (see [7]). Epistemic accessibility relations can be expressed using boolean functions R_i^K: these functions are defined by imposing equivalence on local states.

Figure 1 presents the algorithm for model checking ATEL formulae, based on the parameters presented above. $SAT(\varphi)$ computes the set of global states (expressed as a boolean formula) in which φ holds. The support procedures SAT_X and SAT_K are presented in Figure 2, while SAT_G and SAT_U are defined in a standard way (see also [2, 11] for a similar approach). The main idea is to express SAT_G and SAT_U as fix-point of operators based on SAT_X. The procedure $SAT_X(\varphi, \Gamma)$ uses a double quantification on actions and returns the set of states from which there exists an action for the agents in Γ such that, for all actions of the agents in $\Sigma \setminus \Gamma$, a transition is enabled such that in the next state φ holds. In the support procedures, Act_Γ denotes a joint action performed by group Γ.

3.2.2 Model checking ATOL

The model checking algorithm for ATOL formulae requires to perform evaluation of *uniform strategies* only (see Section 3.1.2). To this end, it is still possible to use the encoding of the parameters into boolean formulae as presented at the beginning of Section 3.2, but the algorithm of Figure 1 needs to be modified. Given an ATOL formula φ and an interpreted system IS, we proceed as follows:

$SAT_X(\varphi, \Gamma)$ {
 Y = $\{g | \exists g'$ s.t. $R_t(g, g')$ and $\exists a \in Act_\Gamma$ s.t.
  $\forall b \in Act_{\Sigma \setminus \Gamma}$, $t(g, \langle a, b \rangle, g')$ and $g' \in SAT(\varphi)\}$
 return Y;
}
$SAT_K(\varphi, i)$ {
 X = $SAT(\neg\varphi)$;
 Y = $\{g \in G$ s.t. $R_i^K(g, g')$ and $g' \in X\}$
 return $\neg$Y;
}

Figure 2: ATEL support procedures

1. We determine the set of agents that appear under any ATOL operator in φ; we denote this set with Δ.
2. We compute all the uniform collective strategies for agents in Δ, denoted with F_Δ.
3. For each uniform strategy $s \in F_\Delta$, we compute the set of reachable global states where agents in Δ adhere to s; we denote this set with G^s. Similarly, we compute new transition relations $R_t^s(g, g')$. We assume that agents not in Δ are allowed *any* strategy, i.e. they are not restricted to uniform strategies.
4. For each $s \in F_\Delta$, we evaluate the formula φ using the parameters G^s and $R_t^s(g, g')$. If the formula is true, we stop the algorithm and we return true.
5. If there is no strategy in F_Δ for which the formula holds, we return false.

Figure 3 summarises the algorithm; notice that the support procedure $SAT2_{ATOL}$ is similar to the procedure in Figure 1. Due to space limitation, we do not report the details of all the support functions, but these can be easily inferred as an extension of the procedures of Figure 2.

Although the number of uniform strategies may grow exponentially, in our algorithm we keep a symbolic representation of the strategies, and we avoid to pre-compute all the parameters. Instead, they are built on-the-fly and the process terminates as soon as a valid strategy is found. Of course, in the worst case this may lead to an exponential number of steps, but this is intrinsic in the nature of ATOL. However, in the experiments presented below, we did not notice big differences in performance between ATEL and ATOL verification.

4 Implementation

In this section we present an implementation of the algorithms presented in Section 3.2. In the implementation, interpreted systems are described using the language ISPL (Interpreted Systems Programming Language). Figure 4 gives a short example of this

$SAT_{ATOL}(\varphi)$ {
 Δ = compute_uniform_agents(φ);
 F_Δ = compute_uniform_strategies(Δ);
 for each ($s \in F_\Delta$) {
  G^s = compute_reachable_states(s);
  R_t^s = compute_transition_relation(s);
  if (I == $SAT2_{ATOL}(\varphi, G^s, R_t^s)$) {
   print ("φ is TRUE");
  }
 }
 print ("φ is FALSE");
}

$SAT2_{ATOL}(\varphi, G^s, R_t^s)$ {
 φ is an atomic formula: return $h(\varphi)$;
 φ is $\neg\varphi_1$: return $G \setminus SAT2_{ATOL}(\varphi_1, G^s, R_t^s)$;
 φ is $\varphi_1 \wedge \varphi_2$: return $SAT2_{ATOL}(\varphi_1, G^s, R_t^s) \cap$
  $SAT2_{ATOL}(\varphi_2, G^s, R_t^s)$;
 φ is $\langle\langle\Gamma\rangle\rangle X\varphi_1$: return $SAT2_{ATOL-X}(\varphi_1, \Gamma, G^s, R_t^s)$;
 φ is $\langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$: return $SAT2_{ATOL-U}(\varphi_1, \varphi_2, \Gamma, G^s, R_t^s)$;
 φ is $\langle\langle\Gamma\rangle\rangle G\varphi_1$: return $SAT2_{ATOL-G}(\varphi_1, \Gamma, G^s, R_t^s)$;
 φ is $K_i\varphi_1$: return $SAT2_{ATOL-K}(\varphi_1, i, G^s, R_t^s)$;
}

Figure 3: ATOL algorithm

```
Agent SampleAgent
 Lstate = {s0,s1,s2,s3};
 Action = {a1,a2,a3};
 Protocol:
   s0: {a1};
   s1: {a2};
   s2: {a1,a3};
   s3: {a2,a3};
 end Protocol
 Ev:
   s2 if ((AnotherAgent.Action=a7);
   s3 if Lstate=s2;
 end Ev
end Agent
```

Figure 4: ISPL example

language. We refer to the files available online for the full syntax of ISPL. ATEL and ATOL formulae to be checked are provided at the end of the specification file, using an intuitive syntax.

MCMAS automatically parses the specification and builds the relevant parameters, stored as OBDD's using the library provided by [25].

We have implemented the two algorithms of Section 3.2. MCMAS is able to recognise whether a formula is an ATEL formula, or an ATOL formula, and calls the relevant function to compute the set of states in which a formula holds, in the form of an OBDD. To determine whether or not a formula holds in the interpreted system, its OBDD is compared with the OBDD representing the set of initial states, and the appropriate output is produced.

MCMAS can be run from the command line, and accepts various options to modify verbosity, to inspect OBDD's statistics and memory usage, to enable variable reordering in the OBDD's (see [25]), etc. These options can be used to determine the "critical" points, and to fine tune the performance of MCMAS.

MCMAS is written in C/C++ and it has been successfully compiled on various platforms, including PowerPC (Mac OS X 10.2 and 10.3), Intel (various Pentium flavours using Linux 2.4 and 2.6), and SPARC (SunOS 5.8 and 5.9). The source code has been compiled with gcc/g++ from version 2.95 till version 3.3.

5 Examples

Although MCMAS is engineered for large systems whose state space may approach 10^{40}, here we exemplify its usage on two small MAS examples. The examples further clarify the subtle differences between ATEL and ATOL.

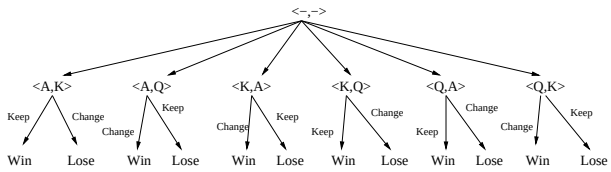

Figure 5: A simple card game

5.1 Card game

This example is presented in [15] and in [17]: an agent (the player) plays a simple card game against another agent, the environment. There are just three cards in the deck: Ace (A), King (K), and Queen (Q); A wins over K, K wins over Q, and Q wins over A. In the initial state no cards are distributed; in the first step, the environment gives a card to the player and takes a card for itself. In the second step, the player can either keep its card, or change it. The game is depicted in Figure 5. A description of this scenario in terms of interpreted systems can be easily obtained, and it is available in the downloadable files. This example illustrates how ATEL semantics differs from the original meaning of ATL operators. Indeed, it is possible to check with MCMAS that the ATEL formula $\langle\langle \text{player} \rangle\rangle F(\text{Win})$ is true in this scenario. As remarked in Section 3.1, the ATEL formula above expresses what player *may* bring about, by randomly selecting the correct move in a global state. In this example, however, it seems more natural to reason with an operator to express the fact that the agent is able to *enforce* a particular state of affairs. This is equivalent to require that the agent has a *feasible* (or uniform) strategy, by using the ATOL formula $\langle\langle \text{player} \rangle\rangle^{\bullet} F(\text{Win})$ As expected, MCMAS reports that the formula is false. This result expresses the fact that the agent does not have a *feasible plan* to enforce a favourable state of affairs.

Our approach on the analysis of plans is slightly different from [16]. Indeed, the evaluation of propositions in interpreted systems is based on states, and there is no machinery to reason about actions in formulae. Hence, we avoid to express concepts such as “knowledge of a strategy”, or “common knowledge of a strategy in a group”, and we see strategies as a property of the model, instead of a group of agents. This is enough to allow model designers to verify the presence of strategies in their models.

5.2 Prisoners’ dilemma

In certain cases, we do not want to reason about *feasible* strategies; instead, we are interested in the analysis of what *may* happen in the worst case scenario. We give an example of this by means of a well known situation: the prisoners’ dilemma.

The dilemma goes as follows: author1 and author2 are arrested, suspected of being the authors of the serious crime of plagiarism. They are kept apart, and the police examines them separately. If they both confess, they will receive five years of jail sentence; if neither confesses, they will receive one year for lack of strong evidence. However, if one confesses but the other does not, the confessor will go free, while the other will receive a ten years jail sentence.

N. crypt.	$\|M\|$	**MOCHA**	MCMAS
3	$7 \cdot 10^{13}$	0.35sec	0.36sec
4	$7 \cdot 10^{18}$	2.32sec	3.8sec

Table 1: Time requirements for 10 formulae.

We are interested here in the following question: is it true that author2 has a way to send author1 to prison?

It is not difficult to describe this scenario with the ISPL language (see the downloadable files). We introduce the proposition author1-prison, which is true if the first author does not go free in the final state. MCMAS correctly reports that the ATEL formula $\langle\langle \text{author2} \rangle\rangle F(\text{author1-prison})$ is true in this interpreted system (notice that, in this case, the equivalent ATOL formula would be true as well).

By extending this simple example to more complex scenarios, ATEL formulae may be used to determine whether or not a component of a system has the power to enforce some unwanted state of affairs. In this kind of analysis we are not interested in the fact that an agent (or a group of agents) has a feasible strategy. Instead, designers typically want to verify that a system is *safe*, irrespective of the (possibly random) choices of some of the components. It seems to us that ATEL offers the proper semantics to express such claims.

6 Experimental results

The examples presented above were checked in less than 0.1sec by MCMAS a standard PC, thus we do not consider them meaningful for evaluating the performance of our tool on bigger examples.

Instead, in this section, we will consider the protocol of the dining cryptographers as presented in [6] to evaluate MCMAS against MOCHA [1], the only model checker available for ATL. We will not introduce the details of the protocol here; for the sake of this paper, it is only important that the example is fairly large, it involves temporal properties, and it can be easily scaled up. We ran the test on a Pentium 4 at 2.8GHz with 1Gbytes of RAM and running Linux (Kernel 2.4.20). The average times required to check ten temporal-only formulae (MOCHA does not support epistemic operators) are reported in Table 1.

We noticed that MOCHA is faster than MCMAS in the construction of the initial parameters: it only takes 1 second to construct a model of the protocol with 4 cryptographers. MCMAS, instead, is slower in constructing the initial OBDD's for the parameters (3.6sec for a similar model construction), but it is actually faster in the verification of formulae. We think that this difference may be due to different implementation choices: MOCHA may compute certain parameters on-the-fly, in the verification process, while we pre-compute the majority of the parameters before the verification process, with the exception of ATOL parameters. Moreover, in MCMAS we build OBDD's for epistemic accessibility relations, and for other parameters that are specific of interpreted systems. This is not done in MOCHA, and might itself be the reason for the difference.

N.Crypt.	$\|M\|$	OBDD's nodes	Memory
3	$\approx 7 \cdot 10^{13}(46)$	$\approx 10^4$	≈ 4.4 MB
4	$\approx 2 \cdot 10^{18}(62)$	$\approx 6 \cdot 10^4$	≈ 5.2 MB

Table 2: Memory requirements.

Finally, we report statistics about the memory usage for the tool. These are reported in Table 2 (in brackets we report the number of boolean variables used).

We see these as encouraging results. MCMAS compares well with MOCHA, a mature model checker for ATL, and has functionalities not supported by it. Moreover, the use of OBDD's allows for a dramatic reduction in the state space, as can been seen when the number of states in the model is compared with the number of OBDD-nodes in Table 2. Furthermore, we believe that the ISPL language that we use is well suited for MAS specification.

7 Conclusion and future work

We have presented two algorithms and a tool for model checking ATEL and ATOL formulae in multi-agent systems. This analysis has prompted us to some comments on the issues related to epistemic extensions of ATL in MAS. We have tested MCMAS with two small examples to ground our claims on ATEL and ATOL. By verifying a fairly large example, we found that MCMAS has a performance comparable to the more mature MOCHA model checker, although it surpasses it in functionality, given its ability to check epistemic formulae.

While we see these as encouraging results, we also acknowledge that more work is required before a mature product is delivered to system designers. In particular, we did not tackle the issue of *fairness constraints* in the verification process, and we did not investigate the complexity of model checking for ATEL and ATOL. It is known [2, 11] that ATL and ATEL have a PTIME model checking algorithm when the model is given explicitly, but the complexity of the problem may grow when we relax certain assumptions on the model (see [24]). On top of this, in our case the model is given *symbolically* in terms of *variables*, and the size of the model could grow exponentially in the number of variables. It would be interesting to evaluate the complexity of ATEL and ATOL model checking when such a symbolic representation is given.

Finally, we did not consider the issue of counter-example generation for unsatisfiable formulae: establishing *why* a formula is false may be, in certain conditions, just as useful as establishing that a formula is true in a model. We leave all these issues for further work.

References

[1] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proceedings of the 10th International Confer-*

ence on Computer Aided Verification (CAV'98), volume 1427 of *LNCS*, pages 521–525. Springer-Verlag, 1998.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[3] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of TACAS'99*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.

[4] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking agentspeak. In J. S. Rosenschein, T. Sandholm, W. Michael, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent systems (AAMAS-03)*, pages 409–416. ACM Press, 2003.

[5] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, 35(8):677–691, 1986.

[6] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[7] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

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

[9] P. Gammie and R. van der Meyden. Mck: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

[10] V. Goranko and W. Jamroga. Comparing semantics for logics of multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[11] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In M. Gini, T. Ishida, C. Castelfranchi, and W. L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 1167–1174. ACM Press, 2002.

[12] G. J. Holzmann. The model checker SPIN. *IEEE transaction on software engineering*, 23(5), 1997.

[13] M. R. A. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, England, 2000.

[14] W. Jamroga. Some remarks on alternating temporal epistemic logic. In B. Dunin-Kęplicz and R. Verbrugge, editors, *Proceedings of the International Workshop on Formal Approaches to Multi-Agent Systems (FAMAS'03)*, pages 133–140, 2004.

[15] W. Jamroga. *Using Multiple Models of Reality. On Agents who Know how to Play Safer.* PhD thesis, University of Twente, Enschede, The Netherlands, 2004.

[16] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 62:1–35, 2004.

[17] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic. Master's thesis, University of Utrech, The Netherlands, 2003.

[18] M. Kacprzak and W. Penczek. A SAT-based approach to unbounded model checking for alternating-time temporal epistemic logic. *Synthese*, 142:203–227, 2004.

[19] A. Lomuscio. *Knowledge Sharing among Ideal Agents*. PhD thesis, School of Computer Science, University of Birmingham, Birmingham, UK, June 1999.

[20] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[21] W. Nabiałek, A. Niewiadomski, W. Penczek, A. Półrola, and M. Szreter. VerICS 2004: A model checker for real time and multi-agent systems. In *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'04)*, volume 170 of *Informatik-Berichte*, pages 88–99. Humboldt University, 2004.

[22] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Knowledge as strategic ability. *ENCTS*, 85(2):1–23, 2003.

[23] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: an algorithm and its implementation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, 2004.

[24] P.Y. Schobbens. Alternating-time logic with imperfect recall. *ENTCS*, 85(2):1–12, 2004.

[25] F. Somenzi. CUDD: CU decision diagram package - release 2.4.0. http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html.

[26] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[27] M. Wooldridge, M. Fisher, M. Huget, and S. Parsons. Model checking multiagent systems with mable. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, Bologna, Italy, July 2002.