Symbolic Model Checking Epistemic Strategy Logic

Xiaowei Huang and Ron van der Meyden
School of Computer Science and Engineering
University of New South Wales, Australia

Abstract

This paper presents a symbolic BDD-based model checking algorithm for an epistemic strategy logic with observational semantics. The logic has been shown to be more expressive than several variants of ATEL and therefore the algorithm can also be used for ATEL model checking. We implement the algorithm in a model checker and apply it to an application on train control system. The performance of the algorithm is also reported, with a comparison showing improved results over a previous partially symbolic approach for ATEL model checking.

Introduction

We are interested in verifying systems in which multiple agents act strategically. ATL (Alur, Henzinger, and Kupferman 2002) generalises the temporal logic CTL (Clarke, Emerson, and Sistla 1986) with *selective quantifications* over the paths, by quantifying over agents' strategy ability. ATEL (van der Hoek and Wooldridge 2002) adds *epistemic operators* into ATL. In (Huang and van der Meyden 2014b; 2014a), an epistemic strategy logic ETLK is shown to be strictly more expressive than several variants of ATEL, e.g., (Schobbens 2004; van Otterloo and Jonker 2005; Jamroga and van der Hoek 2004), etc.

Reasoning about strategies generally needs to be based on the premise that agents have *imperfect information* concerning the underlying state of the system. In this paper, we assume that agents determine their knowledge and behaviour using only their *current observation*.

The main contribution of the paper is a symbolic model checking algorithm for ETLK (and therefore for ATEL), using binary decision diagrams (BDDs). We show that the algorithm performs better than a partially-symbolic approach proposed in (Lomuscio and Raimondi 2006; Busard et al. 2013) for ATEL model checking.

We apply the implemented algorithm on a train control system. The implementation can find strategies for the controller or the trains to follow, so that the system satisfies some critical properties.

An Epistemic Strategy Logic

A distributed or multi-agent system consists of a set $\mathtt{Agt}$ of agents running in an environment. We model these using interpreted systems, following (Fagin et al. 1995). Let $\mathtt{Var}$ be a set of atomic propositions. A *global state* is an element of the set $\mathcal{G} = L_e \times \Pi_{i\in\mathtt{Agt}} L_i$, where L_e is a state of the environment and each L_i is a *local state* for agent i. A *run* is a mapping $r : \mathbb{N} \to \mathcal{G}$ giving a global state at each moment of time. A *point* is a pair (r, m) consisting of a run r and a time $m \in \mathbb{N}$. An *interpreted system* is a pair $\mathcal{I} = (\mathcal{R}, \pi)$, where $\mathcal{R}$ is a set of runs and π is an *interpretation*, mapping each point (r, m) with $r \in \mathcal{R}$ to a subset of $\mathtt{Var}$. For $n \leq m$, we write $r[n \dots m]$ for the sequence $r(n)r(n+1)\dots r(m)$. For each agent $i \in \mathtt{Agt}$, we write $r_i(m)$ for the component of $r(m)$ in L_i, and then define an equivalence relation on points by $(r, m) \sim_i (r', m')$ if $r_i(m) = r'_i(m')$. We also define $\sim_G^D \equiv \cap_{i\in G} \sim_i$, and $\sim_G^C \equiv (\cup_{i\in G} \sim_i)^*$ for $G \subseteq \mathtt{Agt}$.

Let $\mathtt{SVar}$ be a set of variables. The language ETLK($\mathtt{Agt}, \mathtt{Var}, \mathtt{SVar}$) has syntax given by the grammar:

$$\phi \quad ::= \quad p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid A\bigcirc\phi \mid A\Box\phi \mid A(\phi_1 U \phi_2) \mid \exists x.\phi \mid \mathtt{e}_i(x) \mid D_G\phi \mid C_G\phi$$

where $p \in \mathtt{Var}$, $x \in \mathtt{SVar}$, and $G \subseteq \mathtt{Agt}$. The construct $D_G\phi$ expresses that agents in G have distributed knowledge of ϕ, i.e., could deduce ϕ if they pooled their information, and $C_G\phi$ says that ϕ is common knowledge to group G. The temporal formulas $A\bigcirc\phi$, $A\Box\phi$ and $A(\phi_1 U \phi_2)$ have the standard meanings from CTL. The construct $\exists x.\phi$ says that there exists in the system a point x such that ϕ holds at the current point, and $\mathtt{e}_G(x)$ says that agents in G have the same local state at the current point and at the point x. Other operators can be obtained in the usual way, e.g., $\forall x.\phi = \neg\exists x.\neg\phi$, $A\Diamond\phi = \neg E\Box\neg\phi$, etc. We use construct $K_i\phi = D_{\{i\}}\phi$ to express that agent i knows ϕ, and $E_G\phi = \bigwedge_{i\in G} K_i\phi$ to express that every one in G knows ϕ. We write $\mathtt{e}_G(x)$ for $\bigwedge_{i\in G} \mathtt{e}_i(x)$.

A *context* for an interpreted system $\mathcal{I}$ is a mapping Γ from $\mathtt{SVar}$ to global states occurring in $\mathcal{I}$. We write $\Gamma[s/x]$ for the result of changing Γ by assigning global state s to variable x. The semantics of the language ETLK is given by a relation $\Gamma, \mathcal{I}, (r, m) \models \phi$, representing that formula ϕ holds at point (r, m) of the interpreted system $\mathcal{I}$, relative to context Γ. This is defined inductively on the structure of the formula ϕ, as follows:

- $\Gamma, \mathcal{I}, (r, m) \models p$ if $p \in \pi(r, m)$;

- $\Gamma, \mathcal{I}, (r,m) \models \neg\phi$ if not $\Gamma, \mathcal{I}, (r,m) \models \phi$;
- $\Gamma, \mathcal{I}, (r,m) \models \phi \vee \psi$ if $\Gamma, \mathcal{I}, (r,m) \models \phi$ or $\Gamma, \mathcal{I}, (r,m) \models \psi$;
- $\Gamma, \mathcal{I}, (r,m) \models A \bigcirc \phi$ if for all $r' \in \mathcal{R}$ with $r[0 \dots m] = r'[0 \dots m]$, there is $\Gamma, \mathcal{I}, (r', m+1) \models \phi$;
- $\Gamma, \mathcal{I}, (r,m) \models A\Box\phi$ if for all $r' \in \mathcal{R}$ with $r[0 \dots m] = r'[0 \dots m]$, there is $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all $m' \geq m$;
- $\Gamma, \mathcal{I}, (r,m) \models A(\phi U \psi)$ if for all $r' \in \mathcal{R}$ with $r[0 \dots m] = r'[0 \dots m]$, there exists $m' \geq m$ such that $\Gamma, \mathcal{I}, (r', m') \models \psi$ and $\Gamma, \mathcal{I}, (r', k) \models \phi$ for all k with $m \leq k < m'$;
- $\Gamma, \mathcal{I}, (r,m) \models \exists x.\phi$ if $\Gamma[r'(m')/x], \mathcal{I}, (r,m) \models \phi$ for some point (r', m') of $\mathcal{I}$;
- $\Gamma, \mathcal{I}, (r,m) \models \mathtt{e}_i(x)$ if $r_i(m) = \Gamma(x)_i$;
- $\Gamma, \mathcal{I}, (r,m) \models D_G\phi$ if $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all (r', m') such that $(r', m') \sim_G^D (r,m)$;
- $\Gamma, \mathcal{I}, (r,m) \models C_G\phi$ if $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all (r', m') such that $(r', m') \sim_G^C (r,m)$.

Strategic Environment

Since interpreted systems are infinite objects, for purposes of model checking, we work with a finite state object from which an interpreted system is generated. An *epistemic transition system* for agents $\mathtt{Agt}$ is a tuple $M = \langle S, I, \{O_i\}_{i \in \mathtt{Agt}}, \longrightarrow, \pi, \mathcal{F} \rangle$, where S is a set of states, $I \subseteq S$ is a set of initial states, $O_i : S \to \mathcal{O}$ provides agent i with an observation on each state, $\longrightarrow$ is a transition relation (see below), $\pi : S \to \mathcal{P}(\mathtt{Var})$ is a propositional assignment, and $\mathcal{F} = \{Q_1, ..., Q_k\}$ is a generalized Büchi fairness condition such that $Q_i \subseteq S$ for all $1 \leq i \leq k$. A system is said to be finite if all its components, i.e., S, $\mathtt{Agt}$, $\mathtt{Act_i}$ and $\mathtt{Var}$ are finite.

We work with two types of system. In *labelled epistemic transition systems*, or *environments* the transition relation is of type $\longrightarrow \subseteq S \times \mathtt{Act} \times S$, with $\mathtt{Act} = \Pi_{i \in Ags} \mathtt{Act_i}$ a set of joint actions, where each $\mathtt{Act_i}$ is a nonempty set of actions that may be performed by agent i. Intuitively, a joint action $a \in \mathtt{Act}$ represents a choice of action $a_i \in \mathtt{Act_i}$ for each agent $i \in \mathtt{Agt}$, performed simultaneously, and the transition relation resolves this into an effect on the state. We assume that $\longrightarrow$ is serial in the sense that for all states $s \in S$ and actions $a \in \mathtt{Act}$ there exists a state $t \in S$ such that $(s, a, t) \in \longrightarrow$. Once we have selected a strategy for each agent in such a system, we obtain an *unlabeled* epistemic transition system , in which the transition relation is of type $\longrightarrow \subseteq S \times S$, and we assume that this is serial in the sense for each $s \in S$ there exists $t \in S$ such that $s \longrightarrow t$.

A *strategy* for agent $i \in \mathtt{Agt}$ in such an environment M is a function $\alpha : S \to \mathcal{P}(\mathtt{Act_i}) \setminus \{\emptyset\}$, selecting a set of actions of the agent at each state. We call these the actions *enabled* at the state. A strategy α_i for agent i is *uniform* if for all states s, t, if $O_i(s) = O_i(t)$, then $\alpha_i(s) = \alpha_i(t)$. A strategy α_i for agent i is *deterministic* if $\alpha_i(s)$ is a singleton set for all $s \in S$. A strategy $\alpha_G = \langle \alpha_i \rangle_{i \in G}$ for a group G is *locally uniform (deterministic)* if α_i is uniform (respectively, deterministic) for each agent $i \in G$. Given a system M, we write $\Sigma^{unif,det}(M)$ for the set of all deterministic locally uniform joint strategies (and denote other combinations similarly).

We now define an interpreted system that contains all the possible runs generated when agents $\mathtt{Agt}$ behave by choosing a strategy from some set Σ of joint strategies in the context of an environment M. This interpreted system is obtained as the system generated from an unlabeled epistemic transition system that we now define. One innovation, introduced in (Huang and van der Meyden 2014b), is that the construction of this epistemic transition system introduces new agents $\sigma(i)$, for each $i \in \mathtt{Agt}$. Both the observation and the local state of $\sigma(i)$ are the strategy currently being used by agent i. Agent $\sigma(i)$ is not associated with any actions, and is primarily for use in epistemic operators to allow reference to what could be deduced were agents to reason using information about each other's strategies. We can refer, using distributed knowledge operators D_G where G contains the new strategic agents $\sigma(i)$, to what agents would know, should they take into account not just their own observations, but also information about other agent's strategies. For example, $D_{\{i,\sigma(j)\}}\phi$ says that agent i can deduce ϕ from its observation, plus knowledge of the strategy being used by agent j. For $G \subseteq \mathtt{Agt}$, we write $\sigma(G)$ for the set $\{\sigma(i) \mid i \in G\}$. Additionally, we include an agent $\mathtt{e}$ for representing the state of the environment.

Given an environment $M = \langle S, I, \{O_i\}_{i \in \mathtt{Agt}}, \longrightarrow, \pi, \mathcal{F} \rangle$ for agents $\mathtt{Agt}$, and a set Σ of strategies for the group $\mathtt{Agt}$, we define the *strategy space* (unlabeled) epistemic transition system $\mathcal{E}(M, \Sigma) = \langle S^*, I^*, \longrightarrow^*, \{O_i^*\}_{i \in \mathtt{Agt} \cup \sigma(\mathtt{Agt}) \cup \{\mathtt{e}\}}, \pi^*, \mathcal{F}^* \rangle$ for agents $\mathtt{Agt} \cup \sigma(\mathtt{Agt}) \cup \{\mathtt{e}\}$ as follows. The state space is defined by $S^* = S \times \Sigma$, i.e., a state $(s, \alpha) \in S^*$ consists of a state s of M together with a joint strategy α for the set of all agents in M. The transition relation is given by $((s, \alpha), (t, \alpha')) \in \longrightarrow^*$ if $\alpha = \alpha'$ and there exists a joint action a such that $(s, a, t) \in \longrightarrow$ and $a_i \in \alpha_i(s)$ for all agents $i \in \mathtt{Agt}$. Intuitively, in a transition, each agent in $\mathtt{Agt}$ first selects one of the actions enabled by its strategy, and we then make a transition in M using the resulting joint action. There are no changes to the agents' strategies resulting from the transition. The initial states are given by $I^* = I \times \Sigma$, i.e., an initial state consists just of an initial state in M and a choice of joint strategy in Σ. For the observation functions, the definition of O_j^* at a state $(s, \alpha) \in S^*$ depends on the type of j. We define $O_j^*(s, \alpha) = O_j(s)$, for $j \in \mathtt{Agt}$. For $j = \sigma(i)$, with $i \in \mathtt{Agt}$, we define $O_j^*(s, \alpha) = \alpha_i$. For $j = \mathtt{e}$, we define $O_j^*(s, \alpha) = s$. Finally, $\pi^*(s, \alpha) = \pi(s)$ for all states $(s, \alpha) \in S^*$. We let $\mathcal{F}^* = \{Q \times \Sigma \mid Q \in \mathcal{F}\}$.

A *run* of an unlabeled epistemic transition system $\mathcal{E}$ is a sequence $r : \mathbb{N} \to S^*$ such that $r(0) \in I^*$, $(r(k), r(k+1)) \in \longrightarrow^*$ for all $k \in \mathbb{N}$, and for all $1 \leq i \leq k$ there are infinitely many indices $j \geq 0$ for which $O_{\mathtt{e}}^*(r(j)) \in Q_i$. Given an unlabeled epistemic transition system $\mathcal{E}$, we obtain an interpreted system $\mathcal{I}(\mathcal{E}) = (\mathcal{R}, \pi')$ as follows. For a run $r : \mathbb{N} \to S^*$ of $\mathcal{E}$, define the lifted run $\hat{r} : \mathbb{N} \to S^* \times \Pi_{i \in \mathtt{Agt} \cup \sigma(\mathtt{Agt}) \cup \{\mathtt{e}\}} O_i^*(S^*)$, by letting $\hat{r}_i(m) = O_i^*(r(m))$ for $i \in \mathtt{Agt} \cup \sigma(\mathtt{Agt}) \cup \{\mathtt{e}\}$. Then we take $\mathcal{R}$ to be the set of lifted runs $\hat{r}$ with r a run of $\mathcal{E}$. The assignment π' is given by $\pi'(r, m) = \pi^*(r(m))$. Therefore, we start with an environment M and an associated set of strategies Σ, and then work with the language ETLK($\mathtt{Agt} \cup \sigma(\mathtt{Agt}) \cup \{\mathtt{e}\}, \mathtt{Var}, \mathtt{SVar}$) in the interpreted system $\mathcal{I}(\mathcal{E}(M, \Sigma))$.

Given a labeled transition system M, the set Σ of joint strategies, and a formula ϕ of ETLK language, the model checking problem, written as $M, \Sigma \models \phi$, is to decide whether $\mathcal{I}(\mathcal{E}(M, \Sigma)), (r, 0) \models \phi$ for all $r \in \mathcal{R}$.

Embedding of ATEL

The syntax of ATEL is as follows:

$$\phi \quad ::= \quad p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \langle\langle G\rangle\rangle \bigcirc \phi \mid \langle\langle G\rangle\rangle \Box\phi \mid \langle\langle G\rangle\rangle(\phi_1 U \phi_2) \mid D_G\phi \mid C_G\phi$$

where $p \in \mathtt{Var}$ is an atomic proposition and $G \subseteq \mathtt{Agt}$ is a set of agents. For the ATEL semantics of (Schobbens 2004), we have the following translation (Huang and van der Meyden 2014b; 2014a) from ATEL to ETLK. For a formula ϕ, we write ϕ^* for the translation of ϕ, defined inductively on the construction of ϕ by the following rules

$$p^* = p \qquad (\neg\phi)^* = \neg\phi^* \qquad (\phi_1 \vee \phi_2)^* = \phi_1^* \vee \phi_2^*$$
$$(D_G\phi)^* = D_G\phi^* \qquad (C_G\phi)^* = C_G\phi^*$$
$$(\langle\langle G\rangle\rangle \bigcirc \phi)^* = \exists x.E_G(\mathtt{e}_{\sigma(G)}(x) \Rightarrow A \bigcirc \phi^*)$$
$$(\langle\langle G\rangle\rangle \Box\phi)^* = \exists x.E_G(\mathtt{e}_{\sigma(G)}(x) \Rightarrow A\Box\phi^*)$$
$$(\langle\langle G\rangle\rangle \phi_1 U \phi_2)^* = \exists x.E_G(\mathtt{e}_{\sigma(G)}(x) \Rightarrow A(\phi_1^* U \phi_2^*))$$

Intuitively, the construct $\exists x.E_G(\mathtt{e}_{\sigma(H)}(x) \Rightarrow \psi)$ states that there exists a global state x such that every agent in G knows that if the agents in group H run the strategy contained in the global state x, then ψ is guaranteed. The translation applies this with $H = G$ for several ψ.

BDD-based Model Checking Algorithm

We develop a symbolic model checking algorithm for ETLK using Binary Decision Diagrams (BDDs) (Bryant 1986).[1] These are a canonical representation of boolean functions that can, in practice, be quite compact and that support efficient computation of operations, e.g., boolean operations $\wedge, \neg, \vee, \implies, \Leftrightarrow$, boolean quantification $\exists, \forall$, an operation $=$ to decide if two functions are equivalent, and a variable substitution operation $f[v \mapsto v']$ which renames input variable v to v' in the function f. The substitution operation is also lifted to work with sets of variables $f[V \mapsto V']$. These BDD operations allow us to map a boolean or QBF formula to a BDD in an obvious way.

Our model checking algorithm takes a formula representation of the environment M as one of the inputs.

1. Each state s is represented as a boolean *assignment* to the set $\mathtt{Var}$ of atomic propositions. This means that the set of states S can be represented by a boolean formula over $\mathtt{Var}$, and similarly for the set I of initial states, the fairness constraint Q_i for $1 \leq i \leq k$.
2. Actions in $\mathtt{Act_i}$ can be represented as assignments to a set of boolean variables $BAct_i = \{b_{i1}, ..., b_{im_i}\}$, such that $m_i = \lceil \log_2 |\mathtt{Act_i}| \rceil$. Therefore, a joint action a is represented as an assignment to variables $BAct = \bigcup_{i \in \mathtt{Agt}} BAct_i$.
3. For the transition relation $\longrightarrow$, we represent the successor state using "primed" versions of $\mathtt{Var}$, defined by $\mathtt{Var'} = \{v' \mid v \in \mathtt{Var}\}$. The transition relation is presented as a boolean formula over variables $\mathtt{Var} \cup BAct \cup \mathtt{Var'}$, or, in the unlabeled case, over $\mathtt{Var} \cup \mathtt{Var'}$.
4. The observation function O_i is defined by giving a subset $\mathtt{Var_i} \subseteq \mathtt{Var}$ of variables as agent i's observable variables. Therefore, an observation o is an assignment to the set $\mathtt{Var_i}$ of variables. We use formula
$$eq_i \equiv \bigwedge_{v \in \mathtt{Var_i}} v \Leftrightarrow v'$$
to represent that agent i has the same observation on two states that are assignments of $\mathtt{Var}$ and $\mathtt{Var'}$ respectively.

Given a boolean assignment z to a set of variables V, we write $\hat{z}$ for the conjunction of the literals v or $\neg v$, for $v \in V$, such that $z(v) = 1$ or $z(v) = 0$, respectively.

Formulas such as the above can be converted to BDD representations in the usual way. The idea of symbolic model checking is then to compute BDD's representing sets of interest using BDD operations. As an example, we show the computation of reachable states with BDD operations over the above formulas. Let S_{reach} be the set of states reachable via transitions $\longrightarrow$ from a state in I. The BDD representation of this can be computed as the first element g_j such that $g_{j+1} = g_j$, of the following recursively defined sequence:

1. $g_0 = I$, and
2. $g_{i+1} = g_i \vee (\exists \mathtt{Var} \cup BAct : g_i \wedge \longrightarrow)[\mathtt{Var'} \mapsto \mathtt{Var}]$.

Symbolic Representation of Strategies

The key idea of our algorithm is to represent strategies symbolically (rather than explicitly, as in prior work). We focus on uniform and deterministic strategies, where we can take advantage of the fact that agent's choice of actions can be represented using a number of variables that is logarithmic in the size of the agent's action set. The approach can be easily adapted to work with nondeterministic strategies, but requires a linear number of variables in that case.

Let O_i be the set of possible observations of agent i. This can be computed from reachable states, by letting

$$O_i \equiv \{s \restriction \mathtt{Var_i} \mid s \in S_{reach}\}.$$

A uniform and deterministic strategy of agent i can be represented as a function $\alpha_i : O_i \to \mathtt{Act_i}$. To represent the space of such functions, we introduce for each agent $i \in \mathtt{Agt}$ a set of new boolean variables X_i, containing the variables $x_{i,o,j}$ for $o \in O_i$ and $j = 1 \ldots m_i$. Therefore, a strategy α_i for an agent i can be represented by an assignment χ_{α_i} to variables in X_i such that for an observation $o \in O_i$ and $j = 1 \ldots m_i$, we have $\chi_{\alpha_i}(x_{i,o,j}) = 1$ iff $\alpha_i(o)(b_{ij}) = 1$. Let $X = \bigcup_{i \in \mathtt{Agt}} X_i$.

Using this representation of strategies, the formula

$$f_\alpha \equiv \bigwedge_{i \in \mathtt{Agt}} \bigwedge_{o \in O_i} \left(\hat{o} \Rightarrow \bigwedge_{j=1 \ldots m_i} (x_{i,o,j} \Leftrightarrow b_{ij}) \right)$$

over variables $\mathtt{Var} \cup X \cup BAct$ states that each agent i selects its action b from its observation o using the strategy encoded by the assignment to X.

[1] We assume the reader has some familiarity with symbolic model checking, and refer to (Clarke, Grumberg, and Peled 1999) for an exposition of this area.

We also need to represent contexts Γ. A global state in $\mathcal{I}(\mathcal{E}(M,\Sigma))$ corresponds to a pair (s,α) where S is a state of M and α is a joint strategy in M. Thus, for each global state variable $v \in \mathtt{SVar}$ in the formula to be model checked, we introduce a set $X^v = \{x^v \mid x \in \mathtt{Var} \cup X\}$ of variables.

Symbolic Representation of $\mathcal{E}(M,\Sigma^{unif,\mathrm{det}})$

The transition system $\mathcal{E}(M,\Sigma)$ is represented as follows. A state $s^* \in S^*$ is an assignment to variables $\mathtt{Var} \cup X$. The transition relation $\longrightarrow^*$ is represented as a boolean formula over $\mathtt{Var} \cup X \cup \mathtt{Var}' \cup X'$, more specifically

$$\longrightarrow^* \equiv \exists BAct : (\longrightarrow \wedge f_\alpha \wedge f_{eqs})$$

where $f_{eqs} \equiv \bigwedge_{x\in X}(x \Leftrightarrow x')$, represents that the joint strategy is not changed in a run. (Note that the action variables $BAct$ are quantified out here because transitions in $\mathcal{E}(M,\Sigma)$ are unlabelled; there are no actions in this system.)

For the observation function O_i^*, the formula eq_i is identical to that in M for $i \in \mathtt{Agt}$. For $i = \mathtt{e}$, we let $eq_i \equiv \bigwedge_{v\in\mathtt{Var}} v \Leftrightarrow v'$. For $i = \sigma(j)$ for $j \in \mathtt{Agt}$, we let $eq_{\sigma(j)} \equiv \bigwedge_{v\in X_j} v \Leftrightarrow v'$. Moreover, for $v \in \mathtt{SVar}$, we write

$$eq_j^v \equiv \bigwedge_{x\in\mathtt{Var_j}} x \Leftrightarrow x^v \text{ and } eq_{\sigma(j)}^v \equiv \bigwedge_{x\in X_j} x \Leftrightarrow x^v$$

for $j \in \mathtt{Agt}$. Intuitively, eq_j^v states the equivalence of agent j's observations on the global state v and the current state, and $eq_{\sigma(j)}^v$ states the equivalence of X_j variables and X_j^v variables.

The formulas representing states and initial states in $\mathcal{I}(\mathcal{E}(M,\Sigma^{unif,\mathrm{det}}))$ are $S^* = S$ and $I^* = I$, but interpreted over the set of variables $\mathtt{Var} \cup X$, to capture that strategies are unconstrained assignments to the variables X representing uniform deterministic strategies.

Symbolic Algorithm

The BDD-based symbolic algorithm computes recursively, for each subformula φ, a BDD representation of a boolean function S_φ over variables $\mathtt{Var}$, X and X^v such that $v \in \mathtt{SVar}$ occurs in φ, such that $S_\varphi(s) = 1$ iff $\Gamma, \mathcal{I}(\mathcal{E}(M,\Sigma^{unif,\mathrm{det}})), (r,m) \models \varphi$ for some (equivalently, all) points (r,m) such that $r_\mathtt{e}(m) = s$, where the global state $\Gamma(v)$ is encoded by the restriction of the assignment s to X^v. The computation of S_φ can be done recursively as follows. We write $S' = S[\mathtt{Var} \cup X \mapsto \mathtt{Var}' \cup X']$. A state in an unlabeled transition system is *fair* if there is a run from it that satisfies the fairness condition. The sets S_{reach} and S_{fair} of reachable and fair states (respectively) of the unlabeled transition system $\mathcal{E}(M,\Sigma)$ can be computed following techniques from (Clarke, Grumberg, and Peled 1999), as illustrated for S_{reach} above.

Definition 1 *The formula representation of S_φ is computed recursively by*

1. $S_p = p$
2. $S_{\neg\varphi} = \neg S_\varphi$
3. $S_{\varphi_1\wedge\varphi_2} = S_{\varphi_1} \wedge S_{\varphi_2}$
4. $S_{EX\varphi} = ex(S_{fair} \wedge S_\varphi)$
5. $S_{E\Box\varphi}$ *is the first element* U_i *of the sequence* $U_0, U_1, ...$ *such that* $U_i = U_{i+1}$, *where* $U_0 = S_\varphi$ *and* $U_{j+1} = S_\varphi \wedge \bigwedge_{Q\in\mathcal{F}} ex(eu(S_\varphi, U_j \wedge S_Q))$
6. $S_{E(\varphi_1 U \varphi_2)}$ *is the first element* U_i *of the sequence* $U_0, U_1, ...$ *such that* $U_i = U_{i+1}$, *where* $U_0 = S_{\varphi_2} \wedge S_{fair}$ *and* $U_{j+1} = U_j \vee (S_{\varphi_1} \wedge ex(U_j))$
7. $S_{\exists v.\varphi} = \exists X^v : ((S_{fair} \wedge S_{reach})[X \mapsto X^v] \wedge S_\varphi)$
   $S_{\mathtt{e}_G(v)} = \bigwedge_{i\in G} eq_i^v$
8. $S_{D_G\varphi} = \forall \mathtt{Var}' \cup X' : (\bigwedge_{i\in G} eq_i \wedge S'_{fair} \wedge S'_{reach} \Rightarrow S'_\varphi)$
9. $S_{C_G\varphi}$ *is the first element* U_i *of the sequence* $U_0, U_1, ...$ *such that* $U_i = U_{i+1}$, *where* $U_0 = S_\varphi$ *and*

$$U_{j+1} = \bigwedge_{i\in G} \forall \mathtt{Var}' \cup X' : (eq_i \wedge S'_{fair} \wedge S'_{reach} \Rightarrow U'_j)$$

where

- $ex(ss) = \exists \mathtt{Var}' \cup X' : (\longrightarrow^* \wedge ss')$
- $eu(ss_1, ss_2)$ *is the first element* U_i *of the sequence* $U_0, U_1, ...$ *such that* $U_i = U_{i+1}$, *where* $U_0 = ss_2$ *and* $U_{j+1} = U_j \vee (ss_1 \wedge ex(U_j))$

We then have the following characterization of the model checking problem:

Theorem 1 *Let* $M = (S, I, \{O_i\}_{i\in\mathtt{Agt}}, \longrightarrow, \pi, \mathcal{F})$ *be an environment and and* ϕ *an* ETLK *formula. Then we have* $M, \Sigma^{unif,det} \models \phi$ *iff* $\forall \mathtt{Var} \cup X : I^* \wedge S_{fair} \Rightarrow S_\phi$.

There is an exponential expansion from M and $\Sigma^{unif,det}$ to $\mathcal{E}(M,\Sigma^{unif,det})$. The theoretical complexity of the model checking problem is PSPACE-complete (Huang and van der Meyden 2014b; 2014a).

Application: Train Control System

The following train control scenario is a variant of the one presented in (van der Hoek and Wooldridge 2002). There are n trains $T = \{\mathtt{T_1}, ..., \mathtt{T_n}\}$, each of which operates on its own railway. They share a common tunnel that can pass one train at a time. The tunnel has been equipped with a control system C intended to prevent collisions.

Every train $\mathtt{T_i}$ has two one-bit signals, $\mathtt{wsig[T_i]}$ and $\mathtt{esig[T_i]}$, to communicate with the tunnel's control system. The signal $\mathtt{wsig[T_i]}$ is used by the train to notify the control system that it wishes to pass through the tunnel, and the signal $\mathtt{esig[T_i]}$ is used by the control system to notify the train that it is allowed to enter the tunnel. A signal may be in one of two states $\{\mathtt{Green}, \mathtt{Red}\}$. Every train may be in one of the three states $\{\mathtt{Away}, \mathtt{Waiting}, \mathtt{Tunnel}\}$ and the the controller maintains a variable concerning the tunnel that may be in one of the two states $\{\mathtt{Full}, \mathtt{Empty}\}$. Intuitively, the value $\mathtt{Full}$ represents that there *may* be a train in the tunnel. We use $\mathtt{tst[T_i]}$ to denote train $\mathtt{T_i}$'s state and $\mathtt{cst}$ to denote controller's variable. Therefore, we have $\mathtt{Var} = \{\mathtt{cst}\} \cup \{\mathtt{tst[T_i]}, \mathtt{wsig[T_i]}, \mathtt{esig[T_i]} \mid \mathtt{T_i} \in T\}$ and system states are assignments to $\mathtt{Var}$.

The system starts from an initial state where $\mathtt{cst} = \mathtt{Empty}$ and for every train $\mathtt{T_i}$, $\mathtt{tst[T_i]} = \mathtt{Away}$ and $\mathtt{wsig[T_i]} = \mathtt{esig[T_i]} = \mathtt{Red}$. The control system C has two actions $\mathtt{Act_C} = \{\mathtt{Allow}, \mathtt{Skip}\}$ and train $\mathtt{T_i}$ has four actions $\mathtt{Act_{T_i}} = \{\mathtt{Wait}, \mathtt{Enter}, \mathtt{Leave}, \mathtt{Skip}\}$.

We give the transition relation by describing the effects of actions. When the control system C takes action `Allow`, the environment will nondeterministically choose one of the trains $\mathtt{T_i}$ that has sent a request to the control system (i.e., for which $\mathtt{wsig[T_i]} = \mathtt{Green}$), change the tunnel state `cst` into `Full`, turn the signal $\mathtt{esig[T_i]}$ into `Green`, representing that the control system sends a signal to the train $\mathtt{T_i}$ and allows it to enter the tunnel, and turn the signal $\mathtt{wsig[T_i]}$ into `Red`, representing that the request has been accepted. If none of the trains is waiting then variables stay unchanged. We assume that a signal is always received immediately and correctly.

If train $\mathtt{T_i}$ takes action `Wait` when it is not in `Tunnel`, then its own state $\mathtt{tst[T_i]}$ becomes `Waiting` and the signal $\mathtt{wsig[T_i]}$ turns into `Green`, to model that it sends a request to the control system. If train $\mathtt{T_i}$ takes action `Enter`, its own state will be updated into `Tunnel`. Once train $\mathtt{T_i}$ takes action `Leave`, its own state will be updated into `Away`, the variable `cst` will be updated into `Empty`, and the signal $\mathtt{esig[T_i]}$ will be turned into `Red`, representing that this round of passing through the tunnel has finished.

The observation functions are given by letting variables $\{\mathtt{tst[T_i]}, \mathtt{esig[T_i]}\}$ be observable to $\mathtt{T_i}$ and variables $\{\mathtt{cst}\} \cup \{\mathtt{wsig[T_i]} \mid \mathtt{T_i} \in T\}$ be observable to C.

Actions have no effect in situations where they are not legal. For the controller C, action `Allow` is legal when its state `cst` is `Empty`. In all other conditions, action `Skip` is legal. On the other hand, for the train $\mathtt{T_i}$, action `Wait` is legal when it is `Away`, action `Enter` is legal when it is `Waiting`, action `Leave` is legal when it is in `Tunnel`, and action `Skip` is legal at any condition.

We do not need fairness constraints, i.e., $\mathcal{F} = \emptyset$. Several properties are of interest in this system, e.g.,

$$\phi_1 \equiv A\Box \bigwedge_{x,y\in T,\ x\neq y} \neg(\mathtt{tst[x]} = \mathtt{Tunnel} \wedge \mathtt{tst[y]} = \mathtt{Tunnel})$$

which expresses the exclusive access to the tunnel: it is always the case that there do not exist two trains that are in `Tunnel` simultaneously,

$$\phi_2 \equiv \bigwedge_{x\in T} A\Box(\mathtt{tst[x]} = \mathtt{Waiting} \wedge \mathtt{esig[x]} = \mathtt{Green} \Rightarrow A\Diamond\ \mathtt{tst[x]} = \mathtt{Tunnel})$$

which expresses that all trains $\mathtt{T_i}$ will eventually enter the tunnel if they are `Waiting` and have already received the signal $\mathtt{esig[T_i]}$ from the control system, and

$$\phi_3 \equiv \bigwedge_{x\in T} A\Box\, A\Diamond\ \mathtt{tst[x]} \neq \mathtt{Tunnel}$$

which expresses that none of the trains will stay in the tunnel forever. To synthesize trains' collective strategies, we use the following ETLK formula to describe their goals:

$$\exists v E_{\{\mathtt{T_1},\ldots,\mathtt{T_n}\}}(\mathtt{e}_{\sigma(\{\mathtt{T_1},\ldots,\mathtt{T_n}\})}(v) \Rightarrow \phi_1 \wedge \phi_2 \wedge \phi_3) \qquad (1)$$

Note that, to compare the performance of our algorithm with an ATEL model checking algorithm (to be explained later), all formulas we considered in the paper are also expressible in ATEL. For example, the above formula can be expressed as

$$\langle\!\langle\{\mathtt{T_1}, ..., \mathtt{T_n}\}\rangle\!\rangle(\phi_1 \wedge \phi_2 \wedge \phi_3),$$

by the transformation stated before. However, as stated in (Huang and van der Meyden 2014b; 2014a), ETLK has strictly more expressiveness than ATEL.

The algorithm presented in the previous section has been implemented in an epistemic model checker MCK (Gammie and van der Meyden 2004). We describe here some strategies that were discovered by the algorithm.

For the system with two trains, i.e., $T = \{\mathtt{T_1}, \mathtt{T_2}\}$, the algorithm found a strategy for both trains:

$$\alpha_{\mathtt{T_i}} = \begin{cases} \mathtt{Wait} & \text{if } \mathtt{tst[T_i]} = \mathtt{Away} \text{ and } \mathtt{esig[T_i]} = \mathtt{Red} \\ \mathtt{Enter} & \text{if } \mathtt{tst[T_i]} = \mathtt{Waiting} \text{ and } \mathtt{esig[T_i]} = \mathtt{Green} \\ \mathtt{Leave} & \text{if } \mathtt{tst[T_i]} = \mathtt{Tunnel} \text{ and } \mathtt{esig[T_i]} = \mathtt{Green} \\ \mathtt{Skip} & \text{otherwise} \end{cases}$$

That is, the trains repeatedly request to enter the tunnel, enter it when they get the green light to do so, and exit immediately after they have entered the tunnel (when the entry light happens to remain green).

In the following, we consider a different problem, in which we aim to find a strategy for the controller rather than for the trains. For the controller C, both action `Skip` and action `Allow` are legal at all states. On the other hand, we restrict train T_i's behaviour to be to use the strategy given above. (This is done by treating the above behaviour of the trains to be part of the environment.)

In such a system, we are interested in the existence of a strategy for the control system C to follow, so that the whole system can satisfy the properties ϕ_1, ϕ_2 and ϕ_3, expressed as ETLK formula

$$\exists v.K_C(\mathtt{e}_{\sigma(\{C\})}(v) \Rightarrow \phi_1 \wedge \phi_2 \wedge \phi_3) \qquad (2)$$

For the system with two trains, the algorithm found a strategy for the control system: the control system takes action `Allow` if

1. $\mathtt{cst} = \mathtt{Empty}$ and for both trains $\mathtt{T_i}$, we have $\mathtt{wsig[T_i]} = \mathtt{Green}$, or
2. $\mathtt{cst} = \mathtt{Full}$ and one of the trains has $\mathtt{wsig[T_i]} = \mathtt{Red}$ and the other one has $\mathtt{wsig[T_i]} = \mathtt{Green}$.

In other cases, it takes action `Skip`.

The second option is arguably somewhat unexpected. To understand it, we notice that by protocol for the trains, if a train is in tunnel, it will exit the tunnel at the next round. That is, a train will not stay in the tunnel for more than one round. It is therefore safe for the controller to signal to another train that it is safe to enter, since by the time it receives the signal, the tunnel will be empty.

Performance Comparison

In this section, we report the performance of our algorithm on two scalable examples. The experiments were conducted on a Debian Linux system, 3.3 GHz Intel i5-2500 CPU, with each process allocated up to 500M memory.

We compare our algorithm with a partially-symbolic algorithm, also implemented in the same model checker. The algorithm follows the ideas in (Lomuscio and Raimondi 2006; Busard et al. 2013) for ATEL model checking: for a formula $\langle\!\langle G\rangle\!\rangle\phi$, it proceeds by constructing all uniform and deterministic strategies α_G of the agents in G, combining each α_G

No. of Trains	2		3		4	
	size	time	size	time	size	time
fully symbolic, formula (1)	$\|X\| = 24$	1.1	$\|X\| = 36$	35.9	$\|X\| = 48$	813.2
partially symbolic, formula (1)	$\|\alpha_G\| =?$					
fully symbolic, formula (2)	$\|X\| = 7$	1.4	$\|X\| = 14$	2.2	$\|X\| = 26$	13237.2
partially symbolic, formula (2)	$\|\alpha_G\| = 128$	5.7	$\|\alpha_G\| = 16384$	852.1	$\|\alpha_G\| =?$	

Table 1: Strategy Space and Running Times (s) of Train Control System

with the model M to obtain a restricted model $M\alpha_G$, and then applying symbolic model checking on all the models $M\alpha_G$ to check the formula $E_G\phi$.

For the train control system, we scale up the number of trains in the system. The strategy space and running times are given in Table 1. For fully symbolic algorithm, the size of the strategy space is represented as the number of variables in X, while for partially symbolic algorithm, the strategy space is represented as the number of strategies $|\alpha_G|$. The number of strategies $|\alpha_G|$ is computed as the number of times the model checking algorithm is applied on a restricted model $M\alpha_G$ and formula $E_G\phi$. We write $|\alpha_G| =?$ for the experiments which did not terminate in 36 hours.

Note that the fully symbolic algorithm actually works with a number $k(|X|+|\texttt{Var}|)+|X|$ new variables, where k is the depth of quantifier nesting in φ. The X variables represent a strategy space of size $2^{|X|}$. Whereas, in the case of formula (1), where we synthesize a protocol for the trains, the symbolic algorithm of the present paper solves the problem for up to four agents in reasonable time, the partially symbolic algorithm did not terminate even for just two agents. For formula two we are synthesizing a protocol only for the controller, but here also the fully symbolic approach is able to handle larger problems.

Related Work

There are several previous proposals for model checking algorithms for ATEL assuming partial observation and uniform strategies. None of them is fully-symbolic. (Lomuscio and Raimondi 2006) and (Busard et al. 2013) take a similar approach as the partially symbolic algorithm in our experiments, with the latter able to deal with fairness constraints. The algorithm in (Calta, Shkatov, and Schlingloff 2010) is an explicit state algorithm, and involves a computation of all maximal cliques of graphs. The complexity of the algorithm is $O(|\phi| \cdot m \cdot 3^{m/3})$ where m is the number of transitions in the system M.

There is no tool available for model checking ATEL assuming uniform strategies. MCMAS (Lomuscio, Qu, and Raimondi 2009) has an implementation for the semantics of (van der Hoek and Wooldridge 2002), i.e., without considering the uniformity of strategies. It has been argued, in e.g., (Jonker 2003; Schobbens 2004), that uniformity more closely fits the spirit of the epistemic extension, in which observations represent that agents have partial information of the state.

Conclusions

The paper proposes a symbolic BDD-based algorithm for model checking ETLK. We can use the implementation to handle interesting examples, and the experiments show that it can give an improvement of several orders of magnitude over a previous partially-symbolic approach for ATEL model checking.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-Time Temporal Logic. *Journal of the ACM* 49(5):672–713.

Bryant, R. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers* C-35(8):677–691.

Busard, S.; Pecheur, C.; Qu, H.; and Raimondi, F. 2013. Reasoning about strategies under partial observability and fairness constraints. In *1st International Workshop on Strategic Reasoning (SR2013)*, 71–79.

Calta, J.; Shkatov, D.; and Schlingloff, B.-H. 2010. Finding uniform strategies for multi-agent systems. In *Computational Logic in Multi-Agent Systems (CLIMA XI)*, 135–152.

Clarke, E. M.; Emerson, E. A.; and Sistla, A. P. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* 8(2):244–263.

Clarke, E.; Grumberg, O.; and Peled, D. 1999. *Model Checking*. The MIT Press.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning About Knowledge*. The MIT Press.

Gammie, P., and van der Meyden, R. 2004. MCK: Model Checking the Logic of Knowledge. In *Proc. Conf. on Computer-Aided Verification, CAV*, 479–483.

Huang, X., and van der Meyden, R. 2014a. An epistemic strategy logic. In *the 2nd International Workshop on Strategic Reasoning (SR2014)*.

Huang, X., and van der Meyden, R. 2014b. A temporal logic of strategic knowledge. In *the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR2014)*.

Jamroga, W., and van der Hoek, W. 2004. Agents that Know How to Play . *Fundamenta Informaticae* 62:1–35.

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

Lomuscio, A., and Raimondi, F. 2006. Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In *the proceedings of the 5th international joint conference on Autonomous agents and multiagent systems (AAMAS 2006)*, 161–168.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *Proc. Conf. on Computer-Aided Verification*, 682–688.

Schobbens, P.-Y. 2004. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science* 85(2):82–93.

van der Hoek, W., and Wooldridge, M. 2002. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 1167–1174.

van Otterloo, S., and Jonker, G. 2005. On Epistemic Temporal Strategic Logic. *Electronic Notes in Theoretical Computer Science (ENTCS)* 126:77–92.