The Complexity of Model Checking Succinct Multiagent Systems

Xiaowei Huang  
Jinan University, China  
CSE, UNSW, Australia

Qingliang Chen  
Jinan University, Guangzhou 510632, China

Kaile Su  
Jinan University, China  
IIIS, Griffith University, Australia

Abstract

This paper studies the complexity of model checking multiagent systems, in particular systems succinctly described by two practical representations: concurrent representation and symbolic representation. The logics we concern include branching time temporal logics and several variants of alternating time temporal logics.

1 Introduction

Model checking [Clarke et al., 1999] is a promising technique used in the verification of a system implementation against its specification. Taking as inputs a model describing the system implementation and a logic formula characterizing the system specification, a model checking algorithm automatically determines whether the formula is satisfied in the model. Traditional model checking works with temporal logics [Pnueli, 1977; Clarke et al., 1986], which can express properties quantified in terms of time. In particular, branching-time temporal logics CTL and CTL* can express the safety or liveness properties on all or some of the paths from a state.

The research on model checking techniques has been extended to work with systems consisting of multiple interacting agents (or components, processes, etc). The system’s behaviour depends on agents’ strategies, which provide instructions for the agents to make decisions. To characterise these specifications, various logical frameworks on reasoning about strategies have been put forward. In particular, alternating-time temporal logics (ATL and ATL*) [Alur et al., 2002] generalise CTL and CTL* with selective quantifications over the paths, by quantifying agents’ strategy ability.

In a multiagent system, an agent usually has to make decisions based on incomplete information. It is allowed to partially observe the system state and reason about the system or other agents’ behaviours based on the observations. An incomplete information system is particularly suitable for the case where agents have private information which they do not want to be accessed by other agents. In this paper, we assume that agents conduct reasoning based on their current observations. With this assumption, there are two variants$^1$ of ATL and ATL*, namely ATL$_{ir}$ and ATL*$_{ir}$.

$^1$We follow the naming conventions from [Schobbens, 2004].

This paper is to clarify the computational complexities of model checking these logics on multiagent systems. The complexity result of a model checking problem provides a theoretical indication on the scalability of a model checking algorithm. Most of existing works on studying model checking complexity assume an explicit representation of the system implementation by e.g., explicitly enumerating the states and the transition relation, etc. However, this is inconsistent with the ways most of the existing model checkers work.

In the paper, we work with succinct representations of multiagent systems. A succinct representation can be a concurrent representation, where agents run concurrently on their own (explicit) protocols, or a symbolic representation, where agents’ protocols are described by boolean formulas. Concurrent representation has been taken as modelling languages of several model checkers, such as Verics [Kacprzak et al., 2008]. Symbolic representation serves as intermediate structure of most BDD-based or SAT-based symbolic model checkers. Modelling languages of some model checkers, for example NuSMV [Čimatti et al., 2002], MCMAS [Lomuscio et al., 2009] and MCK [Gammie and van der Meyden, 2004], can be translated into a symbolic representation in an obvious way. Therefore, it is more practical to work with succinct representations than explicit representation.

Model checking complexities for explicit representation have been well-understood for the logics that we are interested in. In particular, [Clarke et al., 1986] and [Emerson and Lei, 1987] give the complexities for CTL and CTL*, respectively. [Schobbens, 2004; Jamroga and Dix, 2008] present the complexity for ATL$_{ir}$ and [Schobbens, 2004] shows the complexity for ATL*$_{ir}$.

Model checking complexities for succinct representations are less studied. All complexity results, together with those for explicit representation, are given in Table 1. We prove the results for concurrent representation and symbolic repre-

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>PTIME</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>CTL*</td>
<td>PSPACE</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>ATL$_{ir}$</td>
<td>$\Delta^p$</td>
<td>PSPACE</td>
<td>NEXPTIME</td>
</tr>
<tr>
<td>ATL*$_{ir}$</td>
<td>PSPACE</td>
<td>PSPACE</td>
<td>NEXPTIME</td>
</tr>
</tbody>
</table>

Table 1: Model Checking Complexities
sentations. Some related works will be discussed in relevant sections.

2 Multiagent Systems

As a usual structure stated in [Fagin et al., 1995], a multiagent system consists of a set $Agt$ of agents running simultaneously in an environment. Let $Var$ be a set of atomic propositions. The syntax of the language $ATL^*$ is as follows:

$$
\phi ::= p \mid \neg \phi \mid \phi_1 \vee \phi_2 \mid \langle G \rangle \phi \mid E \phi
$$

$$
\varphi ::= \phi \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid X \varphi \mid \varphi_1 \mathcal{U} \varphi_2
$$

where $p \in Var$ and $G \subseteq Agt$ is a set of agents. Other operators can be obtained in the usual way, e.g., $A\phi = \neg E \neg \phi$, $F\phi = TrueU\phi$, etc. $\phi$ is called state formula and $\varphi$ is called path formula.

The language $CTL^*$ is a sublanguage of $ATL^*$ by removing strategy operator $\langle G \rangle$ from the syntax of $ATL^*$. The language $ATL$ ($CTL^*$) is a sublanguage of $ATL^*$ ($CTL^*$, respectively) by assuming that every path formula $\varphi$ is immediately prefixed with a branching operator $E$ or $A$.

In the following, we will present semantics of the languages on multiagent systems of several different representations.

In a multiagent system, at each time, every agent is in some local state, and the environment is in some environment state. A global state is a collection of environment and local states, one for each agent. At a global state, every agent will make an observation over the system, take a local action and update its local state, and the environment will update the environment state according to the joint local action of the agents.

2.1 Explicit Representation

Let $Act = \Pi_{i \in Agt} Act_i$ be a set of joint actions, where $Act_i$ is a finite set of actions that may be performed by agent $i \in Agt$. We use $O$ to denote the set of all possible observations. A labeled transition system $M$, an explicit representation, consists of a tuple $(S, I, [N_i]_{i \in Agt}, [O_i]_{i \in Agt}, \rightarrow, \pi)$ where $S$ is a set of states, $I \subseteq S$ is a set of initial states, $N_i : S \rightarrow \mathcal{P}(Act_i) \setminus \{\emptyset\}$ assigns each state a nonempty set of legal actions that may be taken by agent $i$, $O_i : S \rightarrow O$ provides agent $i$ with an observation on each state, $\rightarrow \subseteq S \times Act \times S$ is a transition relation, and $\pi : S \rightarrow \mathcal{P}(Var)$ labels each state with a set of atomic propositions. We assume that the transition relation $\rightarrow$ is serial, i.e., for every joint action $a \in Act$ and every state $s$, there exists a state $t$ such that $(s, a, t) \in \rightarrow$.

A (uniform and memoryless) strategy $\theta_i$ of agent $i$ maps each state $s \in S$ to a nonempty set of local actions such that $\theta_i(s) \subseteq N_i(s)$ and for all states $s, t \in S$, $O_i(s) = O_i(t)$ implies $\theta_i(s) = \theta_i(t)$. Further, a strategy $\theta_i$ is deterministic if $\theta_i(s)$ is a singleton set for all $s \in S$. Given a strategy $\theta_i$ for some $i \in Agt$, we write $M\theta_i$ for the system $(S, I, \{\theta_i\} \cup [N_i]_{i \in Agt}, [O_i]_{i \in Agt}, \rightarrow, \pi)$ where the legal action function $N_i$ of agent $j$ is replaced with the strategy $\theta_j$. We say that the agent $j$ follows strategy $\theta_j$ in system $M\theta_i$. For a set $G \subseteq Agt$ of agents, we write $\theta_G = \{\theta_i\}_{i \in G}$ for its collective strategy and $M\theta_G$ for the system where every agent $i \in G$ follows strategy $\theta_i$. Moreover, for any system $M$ in which a (maybe empty) set of agents follow their own strategies, we write $M_0$ for the original system where no strategy has been applied.

A fullpath $\rho$ of $M$ is an infinite sequence of states $s_0s_1\ldots$, such that $\exists a \in Act_1 : (s_i, a, s_{i+1}) \in \rightarrow$ for all $i \geq 0$. We use $\rho(m)$ to denote the state $s_m$ and $\rho[m]$ to denote the suffix starting from $s_m$. Moreover, we write $Path(M, s)$ for the set of fullpaths $\rho$ of $M$ such that $\rho(0) = s$.

The semantics of $ATL^*$ in a labeled transition system $M$ can be entailed by a relation $M, s \models \phi$, inductively defined as follows for state $s \in S$ and formula $\phi$:

- $M, s \models p$ if $p \in Var$ if $p \in \pi(s)$
- $M, s \models \neg \phi$ if not $M, s \models \phi$
- $M, s \models \phi_1 \vee \phi_2$ if $M, s \models \phi_1$ or $M, s \models \phi_2$
- $M, s \models \langle G \rangle \phi$ if there exists a collective strategy $\theta_G$ such that $M_0\theta_G$, $s \models \phi$
- $M, s \models E \varphi$ if there is some $\rho \in Path(M, s)$ such that $M, \rho \models \varphi$

where

- $M, \rho \models \phi$ if $M, \rho(0) \models \phi$
- $M, \rho \models \neg \varphi$ if not $M, \rho \models \varphi$
- $M, \rho \models \varphi_1 \vee \varphi_2$ if $M, \rho \models \varphi_1$ or $M, \rho \models \varphi_2$
- $M, \rho \models X \varphi$ if $M, \rho[1] \models \varphi$
- $M, \rho \models \varphi_1 \mathcal{U} \varphi_2$ if there exists $m \geq 0$ such that $M, \rho[k] \models \varphi_1$, for all $0 \leq k \leq m - 1$, and $M, \rho[m] \models \varphi_2$

Note that, when dealing with formula $\langle G \rangle \phi$, the strategy $\theta_G$ is applied on the original system $M_0$, instead of the current system $M$. Given a labeled transition system $M$ and a formula $\phi$ of some language, the model checking problem is to decide whether $M, s \models \phi$ for all $s \in I$.

For labeled transition systems, the complexity of model checking will be measured over the number $|S|$ of states, the number $|Act|$ of actions, and the size $|\phi|$ of formula. The size of transition relation is polynomial with respect to both $|S|$ and $|Act|$. The size of a formula is measured over the number of modalities it contains.

2.2 Concurrent Representation

A multiagent system can also be defined by specifying the agents and the environment individually. This approach has been adopted by the modeling languages of some model checkers, for example Verics [Kacprzak et al., 2008]. We define a representation that shares common characterisations among them and has sufficient expressiveness. Informally, in a concurrent representation of a multiagent system, every agent and the environment run an individual protocol. At each time, an agent will make an observation over the environment, and then based on the observation and its own current local state, choose a subset of local actions according to its protocol. For every joint action of the agents, the environment will update the state in light of its protocol.

The set $Var$ of atomic propositions is partitioned into disjoint sets $Var_x$ for $x \in Agt \cup \{\epsilon\}$. Let $Obs_x$ be the set of observations which agent $i \in Agt$ can observe from environment
states. The environment \( A_g \) is a tuple \((L_o, I_o, \{P_i\}_{i \in \text{Agt}}, \rightarrow, \pi_o)\), where \( L_o \) is a set of environment states, \( L_o \subseteq L_e \) is a set of initial states, \( P_i \colon L_e \rightarrow \text{Obs}_i \) provides agent \( i \) with an observation on each environment state, \( \rightarrow \subseteq L_e \times \text{Act} \times L_e \) is a transition relation, and \( \pi_o : L_e \rightarrow \mathcal{P}(\text{Var}_e) \) is a labelling function. Note that \([\text{Obs}_i] \leq |L_e|\). The environment has no local action, but may nondeterministically update its own state by taking into consideration the joint actions taken by the agents.

An agent \( A_i \), for \( i \in \text{Agt} \), is a tuple \((L_i, I_i, \rightarrow, \pi_i)\), where \( L_i \) is a set of local states, \( L_i \subseteq L_e \) is a set of initial states, \( \rightarrow \subseteq L_i \times \text{Act} \times L_i \) is a transition relation: a tuple \((l_i, a_i, l'_i) \rightarrow\) means that agent \( i \) is at state \( l_i \) and has an observation \( o_i \) on the environment state, it may take action \( a_i \) and move into the state \( l'_i \). If there are several \( a_i \) with the same \( l_i \) and \( o_i \), the agent \( i \) will nondeterministically choose one of them to execute. A strategy \( \theta_i \) of agent \( i \) can then be redefined as a function mapping from \( L_i \times \text{Obs}_i \) to \( \mathcal{P}(\text{Act}_i) \).

Without loss of generality, we let \( \text{Agt} = \{1, \ldots, n\} \). Given a concurrent representation \( C = \{A_i\}_{i \in \text{Agt}} \cup \{e\} \), we can construct its corresponding labeled transition system \( M(C) = (S, I, \{N_i\}_{i \in \text{Agt}}, \{O_i\}_{i \in \text{Agt}}, \rightarrow, \pi) \) such that

1. \( S = L_e \times \prod_{i \in \text{Agt}} L_i \times I_i \times \prod_{i \in \text{Agt}} L_i \),
2. for all states \( s \triangleq (l_e, l_1, \ldots, l_n), O_i(s) = (P_i(l_i), l_i) \),
3. for all states \( s \triangleq (l_e, l_1, \ldots, l_n) \in S \) and all \( a_i \in \text{Act}_i \), we let \( a_i \in N_i(s) \) if and only if there exists a local state \( l'_i \in L_i \) such that \((l_i, P_i(l_i), a_i, l'_i) \rightarrow\) there, \( \rightarrow \triangleq \{(l_i, a_i, l'_i) \rightarrow\} \),
4. for all states \( s \triangleq (l_e, l_1, \ldots, l_n) \), \( s' \triangleq (l'_e, l'_1, \ldots, l'_n) \) and joint actions \( a \equiv (a_1, \ldots, a_n) \), we have that \((s, a, s') \rightarrow\) if and only if \((l_i, a_i, l'_i) \rightarrow\) for all agents \( i \in \text{Agt} \), there is \( l_i \neq l'_i \),
5. \( \pi(s) = \bigcup_{i \in \text{Agt}} \pi_i(l_i) \).

The model checking problem is, given a concurrent representation \( C \) and a formula \( \phi \), to decide whether \( M(C) \models \phi \). The complexity is measured by the number \(|\bigcup_{i \in \text{Agt}} L_i| \) of local states, the number \(|\bigcup_{i \in \text{Agt}} \text{Act}_i| \) of local actions, and the size of formula \( \phi \).

### 2.3 Symbolic Representation

To conduct model checking, most BDD-based or SAT-based model checkers transform a multiagent system described by a modeling language into a certain form of symbolic representation, which uses propositional formulas to represent system components, e.g., the set of initial states, the transition relation, etc. Here we take a usual form of symbolic representation which are expressive enough to describe multiagent systems. Also, we note that modeling languages of some model checkers, e.g., NuSMV [Cimatti et al., 2002], MCMAS [Locusscino et al., 2009] and MCK [Gammanie and van der Meyden, 2004], can be converted into such a representation linearly in an obvious way.

The general idea for a symbolic representation comes from the fact that a formula can be taken to represent a set of states or a transition relation. Every truth assignment over a set of state variables can be regarded as a state. Therefore, a formula over the set of state variables represents the set of states whose corresponding assignments satisfy the formula. Furthermore, a transition from a state to a next state by taking an action can be a truth assignment to the union set of state variables, action variables, and next-time state variables. A formula over the union set of variables can then represent a set of transitions, i.e., a transition relation.

Given a set \( V \) of atomic propositions, we let \( V' = \{v' \mid v \in V\} \) be the set of next-time variables of \( V \), and write \( \mathcal{B}(V) \) to be the set of propositional formulas over \( V \). Local actions \( \bigcup_{i \in \text{Agt}} \text{Act}_i \) can be regarded as atomic propositions. The environment \( \text{Agt}_e \) is a tuple \((\text{Var}_e, \text{Ini}_e, \text{Trn}_e)\), where \( \text{Var}_e \) is a set of environment variables, formula \( \text{Ini}_e \in \mathcal{B}(\text{Var}_e) \) represents a set of initial states, formula \( \text{Trn}_e \in \mathcal{B}(\text{Var}_e \cup \text{Var}_e') \cup \bigcup_{i \in \text{Agt}} \text{Act}_i \) represents a transition relation of the environment. We assume that the propositional formulas are of size polynomial with respect to the variables. This assumption also applies to the agents.

An agent \( \text{Act}_i \) is a tuple \((\text{Var}_i, \text{OVar}_i, \text{Ini}_i, \text{Trn}_i)\), for \( i \in \text{Agt} \), where \( \text{Var}_1 \) is a set of local variables, \( \text{OVar}_1 \subseteq \text{Var}_e \) is a set of environment variables that are observable to agent \( i \), formula \( \text{Ini}_i \in \mathcal{B}(\text{Var}_i) \) represents a set of initial states, formula \( \text{Trn}_i \in \mathcal{B}(\text{Var}_i \cup \text{Var}_i' \cup \text{Var}_1 \cup \bigcup_{i \in \text{Agt}} \text{Act}_i) \) represents a transition relation for agent \( i \). For a set \( V \) of variables and a formula \( f \) over \( V \), we write \( sa(V, f) \) for the set of satisfiable assignments of \( V \) on \( f \). A strategy \( \theta_i \) of agent \( i \) can then be redefined as a boolean formula over the variables \( \text{OVar}_1 \cup \text{Var}_1 \cup \text{Act}_i \).

Given a symbolic representation \( F = \{\text{Agt}_i\}_{i \in \text{Agt}} \), we can construct its corresponding labeled transition system \( M(F) = (S, I, \{N_i\}_{i \in \text{Agt}}, \{O_i\}_{i \in \text{Agt}}, \rightarrow, \pi) \) such that

1. \( S = \text{sa}(\text{Var}, \text{True}), I = \text{sa}(\text{Var}, \text{Ini}_i \wedge \bigwedge_{i \in \text{Agt}} \text{Ini}_i) \),
2. for all states \( s \in S \) and all \( a_i \in \text{Act}_i \), we let \( a_i \in N_i(s) \) if and only if \( \text{Trn}_i \wedge s \wedge a_i \neq \text{False} \),
3. \( \rightarrow \triangleq \text{sa}(\text{Var} \cup \text{Var}' \cup \bigcup_{i \in \text{Agt}} \text{Act}_i \cup \text{Trn}_i \wedge \bigwedge_{i \in \text{Agt}} \text{Trn}_i) \),
4. for all states \( s \in S \), \( O_i(s) = s \upharpoonright (\text{OVar}_1 \cup \text{Var}_1) \), and
5. \( p \in \pi(s) \) if and only if \( p \in s \).

The model checking problem is, given a symbolic representation \( F \) and a formula \( \phi \), to decide whether \( M(F) \models \phi \). The complexity is measured over the number \(|\text{Var}|\) of atomic propositions, the number \(|\bigcup_{i \in \text{Agt}} \text{Act}_i| \) of local actions, and the size \(|\phi|\) of formula.

### 3 Complexity Results for Incomplete Information Systems

Now we are ready to investigate the complexities of model checking logics in the two succinct representations of multiagent systems.

#### 3.1 Concurrent Representation

Our concurrent representation of a multiagent system is different with the concurrent programs in [Kupferman et al., 2000]. The concurrent representation is based on the idea of synchronous languages (e.g. Estrel and Lustre) which have been widely used in modeling reactive systems. On the other hand, concurrent programs base the idea on process algebraic
languages, which have been extensively studied for modelling asynchronous processes.

A concurrent program can be seen as a complete information multitagent system where there exists no environment and agents synchronise their behaviours by taking the same action. Each agent $i$ has a set of legal action $\Act_i$ which may be overlapping, and the product system has actions $\bigcup_{i \in \Act} \Act_i$. A tuple $(s, a, t)$ is a transition between states $s \equiv (l_1, \ldots, l_n)$ and $t \equiv (l'_1, \ldots, l'_n)$, if for all $i \in \Act$, $1) a \in \Act_i$ implies $(l_i, a, l'_i) \rightarrow s$, and $2) a \notin \Act_i$ implies $l_i = l'_i$. Due to different constructions, we can not directly derive complexity results of concurrent representation from those of non-concurrent programs.

[Jamroga and Agotnes, 2007] investigates modular interpreted systems (MIS), in which agents take actions by considering the influence emitted by other agents. For ATL$_\mu$ and ATL$_\mu^*$ model checking, an MIS will be unfolded into different explicit representations. They conjectured that although ATL$_\mu^*$ model checking is easier than that of ATL$_\mu$ in explicit representation, it is harder in MIS. This is different with our results which are based on multitagent systems.

**Theorem 1** The complexity of model checking CTL is PSPACE-hard for the concurrent representation of multitagent systems.

**Proof:** We proceed by a reduction from the problem of accepting an empty input tape on linear bounded automata (LBA). A nondeterministic Turing machine (NTM) $T$ is a tuple $(Q, \Gamma, \delta, q_0, F)$ where $Q$ is a finite set of states, $\Gamma$ is a finite set of alphabets including a special blank symbol $b$, $\delta : Q \times \Gamma \rightarrow 2^P(Q \times \Gamma \times \{-1, 1\})$ is the transition relation, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. Intuitively, a transition $(q, a, q', a', d)$ means that when the machine is at state $q$ and reads $a$ from the current tape cell, it will transit to state $q'$, write $a'$ to the current cell, and move its reading head to one of the neighbour cells in the direction $d$. The head moves left, if $d = -1$, and moves right, if $d = 1$.

We define the size of a Turing machine as the size of space needed to record its transition relation, i.e., $2 \times |\Gamma| \times |Q|^2$. An LBA is an NTM which uses $n$ tape cells for a Turing machine description of size $n$. It is well known that the following problem is PSPACE-complete: given an LBA, to decide whether there exists a computation that accepts empty tape.

We let $\Act = \{1, \ldots, n\}$ such that each tape cell is controlled by an agent. For $i \in \Act$, we define $i \oplus 1 = i + 1$, if $i < n$, and $i \oplus 1 = 1$, if $i = n$. Moreover, $i \ominus 1 = i - 1$, if $i > 1$, and $1$, otherwise. Let $\Act_i = \{\tau\} \cup \{act_{a} : a \in \Gamma \setminus \{b\}\}$ for $i \in \Act$. We write $a_i$ for agent $i$’s local action in the joint action $a$. The environment $A_e$ is $(L_e, I_e, (P_i)_{i \in \Act}, \rightarrow_e, \pi_e)$ such that

1. $L_e = Q \times \{1, n\}$, i.e., an environment state records the machine state $q$ and the current reading head position,
2. $I_e = \{(q_0, 1)\}$, i.e., initially, the machine is at initial state and the reading head is at the leftmost position,
3. $P_e(s) = s$, i.e., agents can see the environment state,
4. for all $(q, a, q', a', d) \in \delta$, we let $(q, m), ja(a, q', m \oplus d) \rightarrow_e$ for all $1 \leq m \leq n$, if $ja_m = act_{a}$ and $ja_k = \tau$ for all $k \neq m$, and
5. $acc \in \pi_e((q, m))$ for all $1 \leq m \leq n$, if $q \in F$.

Let $\text{Obs}_e = L_e$. Agent $A_i$ is $(L_i, I_i, \rightarrow_i, \pi_i)$ such that

1. $L_i = \Gamma$, i.e., the agent records the symbol on its cell,
2. $I_i = \{b\}$, i.e., the agent starts with the blank symbol,
3. the transition relation $\rightarrow_i$ includes
   
   (a) $(a, (q, i), act_{a}, a')$ for all $(q, a, q', a', d) \in \delta$, and
   (b) $(a, (q, k), \tau, a)$ for all $q \in Q$ and $k \neq i$.
4. $\pi_i(l) = \emptyset$ for all local state $l \in L_i$.

To see how the system $C = \{A_i\}_{i \in \Act} \cup \{\delta\}$ simulates the computation of the LBA, we first see that $((q_0, 1), b, \ldots, b)$, the single initial state of $M(C)$, corresponds to the initial configuration of the machine $T$ that it is at state $q_0$, the reading head resides at position 1, and the tape is empty. Then for any state such that $l_e = (q, m)$ and $l_m = a$, if there is a transition $(q, a, q', a', d) \in \delta$ then by the construction, agent $A_m$ will transit into state $a'$ and execute the action $act_{a}$. Other agents $A_k$ for $k \neq m$ will execute $\tau$ action and stay at the same state. The environment will respond to the joint action by transiting into state $q'$ and moving the reading head to the position $m \oplus d$.

Therefore, the existence of a computation to accept the empty tape (i.e., reach an accepting state) is equivalent to the model checking problem $M(C) \models \phi$ acc.

□

**Theorem 2** The complexity of model checking ATL$_\mu^*$ is in PSPACE for the concurrent representation of multitagent systems.

**Proof:** We present a PSPACE model checking algorithm for ATL$_\mu^*$. To decide if $M(C) \models \phi$, the algorithm returns the reversed result of the following procedure:

1. guesses an initial states $s_0$ of the model $M(C)$ and
2. returns the reversed result of $\text{sat}(C, s_0, \phi)$.

The function $\text{sat}(C, s, \phi)$ is computed inductively as follows.

- $\text{sat}(C, s, p)$ for $p \in \text{Var}$ if $p \in \pi(s)$
- $\text{sat}(C, s, \neg \phi)$ if not $\text{sat}(C, s, \phi)$
- $\text{sat}(C, s, \phi_1 \lor \phi_2)$ if $\text{sat}(C, s, \phi_1)$ or $\text{sat}(C, s, \phi_2)$
- $\text{sat}(C, s, \langle \text{G}(\phi) \rangle \phi)$ is the result of guessing a strategy $\theta_G$ and then verifying $\text{sat}(C_0[\theta_G], s, \phi)$, where $C_0[\theta_G]$ is a system by updating every agent $i \in G$’s transition relation $\rightarrow_i$ to make it consistent with the strategy $\theta_i$ in the original system $C_0$.
- $\text{sat}(C, s, \langle \text{E} \phi \rangle \phi)$ if $\text{psat}(C, s, \phi)$.

The function $\text{psat}(C, s, \phi)$ is computed via the automata theoretic approach for LTL model checking [Vardi and Wolper, 1986], whose idea is to reduce the model checking problem into the language emptiness problem of the product Büchi automaton $M(C) \times A_\varphi$, where $A_\varphi$ is the Büchi automaton for the formula $\varphi$. Note that, we use $A_\varphi$, instead of the usual $A_\varphi$ in LTL model checking, because $\varphi$ comes from formula $E \varphi$. The sizes of the automaton $A_\varphi$ and the system $M(C)$ are exponential with respect to $\varphi$ and $C$, respectively. However, we do not need to construct them (and the product automaton) explicitly. Instead, we treat the emptiness check as a Savitch-style search [Savitch, 1970] by a nondeterministic procedure which takes polynomial size of space. We omit the details.
of the search algorithm because it is a simple adaptation to the standard automata theoretic approach [Vardi and Wolper, 1986].

The nondeterministic search algorithm on \( M(C) \times A_e \) involves the evaluations of state subformulas \( \psi \) of \( \varphi \) over the states of \( M(C) \). These evaluations can be done inductively by taking the procedure \( sat(C, s, \varphi) \).

Let \( nL = |\bigcup_{i \in \text{At}_{i}} L_i| \) and \( nA = |\bigcup_{i \in \text{At}_{i}} A_{c,t,i}| \). To handle state formulas, the algorithm needs to remember the current state, which takes \( O(|\text{At}| \times \log nL) \) bits, the current strategy \( \theta \), which takes up to \( \sum_{i \in \text{At}_{i}} |L_i| \times |A_{c,t,i}| = O(|\text{At}| \times nL \times nA) \) bits, and the current formula, which takes up to \( O(|\varphi|) \) bits of space. To handle path formulas, the algorithm needs up to \( O(|\text{At}| \times \log nL + |\varphi|)^2 \) bits of space for the Savitch-style search. Therefore, the space requirement is \( sp = O(|\text{At}| \times \log nL + |\varphi|)^2 + |\text{At}| \times nL^2 \times nA \).

The algorithm uses \( at = O(|\varphi|) \) number of alternations. By Theorem 4.2 of [Chandra et al., 1980], the algorithm can be simulated by a deterministic machine using space \( ar \times sp + sp^2 \), which is polynomial with respect to \( |\text{At}|, nL, |\varphi|, \) and \( nA \). Therefore, it is in PSPACE.

The above theorems lead to the following conclusions.

**Corollary 1** The complexities of model checking CTL, CTL*, ATL, ATL* are all PSPACE-complete for the concurrent representation of multiagent systems.

**Proof:** The lower bounds are obtained by Theorem 1 and the fact that CTL is a sublanguage of all other languages. The upper bounds are obtained by Theorem 2 and the fact that all other languages are subsumed by ATL*.

3.2 Symbolic Representation

Now we move on to examine the complexity on symbolic representation. As will be shown, the complexities for CTL and CTL* are the same with those on concurrent representation. However, the complexities for ATL and ATL* are higher than those on concurrent representation.

**Theorem 3** The complexities of model checking CTL and CTL* are PSPACE-complete for the symbolic representation of multiagent systems.

**Proof:** The lower bound is obtained by a reduction from concurrent representation. Let \( C = \{ A_{i}, \varphi \in \text{At}_{i}, \varphi_{i} \} \). We introduce two boolean variables \( b_x \) and \( b'_x \) for each state \( s \in \bigcup_{i \in \text{At}_{i}} L_{t,i} \) and one boolean variable \( b_o \) for each observation \( o \in \text{Obs}_s \) with \( i \in \text{At} \). We define several formulas:

1. \( f_{s,s} = b_s \land \bigwedge_{i \in L_{t,i}} \neg b_i \), expressing that \( s \) is the current state of \( x \in \text{At} \lor \varphi \), and \( f_{s,s} = b'_s \land \bigwedge_{i \in L_{t,i}} \neg b'_i \), expressing that \( s \) is the next-time state of \( x \in \text{At} \lor \varphi \).
2. \( g_{i,o} = b_o \land \bigvee_{i \in \text{Obs}_s} \neg b_{i,o} \), expressing that the current observation of agent \( i \) on the environment state is \( o \), and
3. \( h_i = \bigvee_{s \in L_{i}} (f_{s,s} \land g_{i,o}) \), expressing the function \( P_i \) by the relation between states and observations.

For a joint action \( a \equiv (a_1, \ldots, a_n) \), we let \( k_a = \bigwedge_{i \in \text{At}_{i}} a_i \). We construct \( \text{Ag}_{t,e} = (\text{Var}_{e}, \text{Ini}_{e}, \text{Trn}_{e}) \) such that

1. \( \text{Var}_{e} = \{ b_i \mid s \in L_e \} \cup \bigcup_{i \in \text{At}_{i}} \{ b_{i,o} \mid o \in \text{Obs}_s \} \),
2. \( \text{Ini}_{e} = \bigwedge_{i \in \text{At}_{i}} h_i \land \bigwedge_{s \in L_e} f_{s,s} \land \) and
3. \( \text{Trn}_{e} = \bigwedge_{i \in \text{At}_{i}} h_i \land \bigvee_{s \in L_{i}} f_{s,s} \land k_a \land f'_{s,s} \).

Intuitively, in \( \text{Trn}_{e} \), the formula \( \bigvee_{s \in L_{i}} f_{s,s} \land k_a \land f'_{s,s} \) encodes all possible transitions, and then the formula \( \bigwedge_{i \in \text{At}_{i}} h_i \) tells the observations of agents. It is similar for \( \text{Ini}_{e} \).

Moreover, we have \( \text{Ag}_{t} = (\text{Var}_{t}, 0 \text{Var}_{t}, \text{Ini}_{t}, \text{Trn}_{t}) \) such that

1. \( \text{Var}_{t} = \{ b_i \mid s \in L_t \} \), \( 0 \text{Var}_{t} = \{ b_o \mid o \in \text{Obs}_s \} \),
2. \( \text{Ini}_{t} = \bigwedge_{i \in \text{At}_{i}} h_i \land \bigwedge_{s \in L_t} f_{s,s} \land \) and

From the way of constructing its explicit representation, a global transition \( \rightarrow \) will need to have the same \( g_{i,o} \) on both \( \text{Trn}_{t} \) and \( \text{Trn}_{e} \). It reflects the fact that agent makes an observation on the environment state, and then the observation is taken into consideration when making local transition.

The symbolic representation \( F = \{ \text{Ag}_{t}, \text{Ag}_{t,e}, \text{Ini}_{t}, \text{Trn}_{t} \} \) is of size polynomial with respect to \( C \), and the above construction can be done in polynomial time. Also, it is not hard to see that \( M(C) \models \varphi \) if and only if \( M(F) \models \varphi \).

The upper bound can be obtained by reusing the algorithm in Theorem 2. We only describe the differences. First, the procedure for \( sat(C, s, \langle G \rangle \varphi) \) is removed and therefore the algorithm needs only a constant number, instead of a polynomial number, of alternations. Second, during the Savitch-style search for path formulas, the guessing of states can be done in polynomial time by guessing the value for each variable in \( \text{Var} \). Third, the evaluation of transitions between states are done by guessing a joint action and then evaluating the satisfiability of the boolean formula \( \text{Trn}_{e} \land \bigwedge_{i \in \text{At}_{i}} \text{Trn}_{t} \), which by definition is in polynomial size.

Therefore, the complexity is in PSPACE because the algorithm can be implemented by a nondeterministic machine with polynomial space.

**Theorem 4** The complexities of model checking ATL and ATL* are NEXPTIME-complete for the symbolic representation of multiagent systems.

**Proof:** The lower bound can be obtained by a reduction from satisfiability of dependency quantified boolean formulas (DQBF) [Peterson et al., 2001]. Let \( X_1, \ldots, X_n, Y_1, \ldots, Y_n \) be tuples of boolean variables and \( F(X_1, \ldots, X_n, Y_1, \ldots, Y_n) \) be a boolean formula over these variables. A DQBF formula can be written as

\[
\forall X_1 \ldots \forall X_n \exists Y_1(X_1) \ldots \exists Y_n(X_1, \ldots, X_n) : F(X_1, \ldots, X_n, Y_1, \ldots, Y_n).
\]

Intuitively, the formula requires that the values of variables \( Y_1 \) depend only on the values of \( X_1 \), the values of \( Y_2 \) depend only on the values of \( X_1 \) and \( X_2 \), and so on. More precisely, such a formula is satisfiable if there exist tuples of boolean expressions \( g_i(X_i) \) (in variables \( X_i \)) through \( g_n(X_1, \ldots, X_n) \) (in variables \( X_1, \ldots, X_n \)) such that the QBF formula

\[
\forall X_1 \ldots \forall X_n (F(X_1, \ldots, X_n, g_1(X_1), \ldots, g_n(X_1, \ldots, X_n)))
\]

is True. It has been shown that every QBF formula can be expressed as a DQBF formula, and the satisfiability problem of DQBF is NEXPTIME-complete [Peterson et al., 2001].

Given a DQBF formula, we construct a symbolic representation. Let \( X = X_1 \cup \ldots \cup X_n \) and \( Y = Y_1 \cup \ldots \cup Y_n \). The system consists of a set of agents \( \text{Ag} = Y \). Every agent decides
the value of a variable from some $Y_k$ based on the values of the
variables $X_1, \ldots, X_k$, which are made observable. Agent
$y \in Y$ has two actions, i.e., $\text{Act}_y = \{ \text{set} T_y, \text{set} F_y \}$.

The environment represents the $X$ and $Y$ variables and han-
dles the evaluation of the formula $F$. More specifically, we have $\text{Agt}_e = (\text{Var}_e, \text{Ini}_e, \text{Trn}_e, \text{set}_e)$ such that

1. $\text{Var}_e = X \cup Y \cup \{ f \}$, $\text{Ini}_e = \text{True}$,
2. $\text{Trn}_e = \wedge_{y \in Y} (\text{set} T_y \implies \text{y'}) \land \text{set} F_y \implies \neg \text{y'}) \land (f \iff F(X_1, \ldots, X_n, Y_1, \ldots, Y_n),

where $Y'_j = \{ y' \mid y \in Y_j \}$ for $1 \leq j \leq n$. That is, the
environment sets the next-time value of each variable $y' \in Y$
to true if the corresponding agent is performing the action
$\text{set} T_y$. The value of the formula $F$, assigned to the vari-
able $f$, is then computed by taking the next-time values.

For every $k = 1, \ldots n$ and variable $y \in Y_k$, we have agent
$\text{Agt}_y = (\text{Var}_y, \text{Ini}_y, \text{Trn}_y, \text{set}_y)$ such that

1. $\text{Var}_y = \emptyset$, $\text{Var}_y = X_1 \cup \ldots \cup X_k$ consists of the set of $X$
variables on which $y$ may depend,
2. $\text{Ini}_y = \text{True}$, and $\text{Trn}_y = \text{set} T_k \lor \text{set} F_y$.

Intuitively, every agent is attached with a variable, and an
agent observes the variables on which the value of its variable
depends and then makes decision on the value of its variable.

Therefore, it is straightforward to show that the satisfiability
of DQBF formula is equivalent to decide whether $M(F) \models (Y) \forall X f$.

For the upper bound, we can reuse the algorithm in The-
orem 2, with some changes to obtain a different complexity.

One of the significant changes exists in dealing with strategy
formulas. A strategy $\theta_i$ may be represented by giving the
truth table for the formula $\beta(\theta)$, where the input variables are
$0 \text{Var}_i \cup \text{Var}_i \cup \text{Act}_i$. Each time when dealing with formula
$\beta(G) \forall \theta$, the algorithm nondeterministically guesses this truth
table representation of $\theta_i$ for every $i \in G$. This phase takes
exponential time.

To handle path formulas $\phi$, we explicitly construct the
product automaton $M(F) \times A_\phi$, which is of exponential size
with respect to both $F$ and $\phi$. Note that, we do not explic-
itsly construct $F_0[\beta_i]$. Instead, we will look up the truth
table when evaluating a transition relation. The checking of the
emptiness of a Büchi automaton can be done in polynomial
time [Vardi and Wolper, 1986].

Finally, because the number of alternation is polynomial,
the algorithm can be implemented with a nondeterministic
machine in exponential time, i.e., in NEXPTIME.

4 Conclusion and Future Work

This paper presents complexity results for model checking
several logics (CTL, CTL*), ATL, ATL* on two succinct represen-
tations of multagent systems. For concurrent repre-
sentation, it is shown that all of them are PSPACE-complete.

On the other hand, for symbolic representation, the complex-
ities for branching time logics remain at PSPACE-complete,
while they are NEXPTIME-complete for ATL and ATL*.

The reason for this increase is that the size of a strategy is
exponential for symbolic representation.

The increase of computational complexity from PSPACE
to NEXPTIME for symbolic representation reflects the actual
situation that it is hard to find an efficient symbolic algorithm
for ATL and ATL*. There are only a few attempts. In [Lo-
muscio and Raimondi, 2006], an algorithm is proposed to first
explicitly enumerate all possible strategies for a group and
then for every strategy, applying symbolic algorithm for CTL
over the system updated with that strategy. This work is later
extended with the capability of handling fairness constraints
in [Busard et al., 2013]. Because the number of strategies can be
exponential over the number of system states (and local ac-
tions if considering nondeterministic strategies), the explicit
enumeration of strategies can not make the algorithms scale
well in practical examples. In [Huang and van der Meyden,
2014b], a fully symbolic algorithm is proposed to tackle this
situation. The general idea is to have a symbolic encoding
of the strategy space, and then take advantage of the space-
efficiency of BDDs in achieving a succinct encoding of the
product system. The experimental results show a significant
improvement over the previous approach. A similar idea is
also presented in [Cermák et al., 2014] independently for a
slightly different logic.

For the future work, we may study the complexity for log-
ics with richer expressiveness, e.g., [Huang and van der Mey-
den, 2014c; 2014a; Cermák et al., 2014], or different memory
requirements such as perfect recall, where agents have mem-
ory to remember all past observations, or clock semantics,
where agents can observe a common global clock value. For
semantics with memory, we mention existing complexity re-
results for explicit representation [van der Meyden and Shilov,
1999; Huang and van der Meyden, 2010; Guelev et al., 2011;
Huang, 2015].

We are also interested in the complexity for succinct rep-
resentations of complete information systems. Complete in-
formation systems can be seen as special cases of incomplete
information systems, such that agents can observe the under-
lying system state. It is therefore reasonable to expect that the
complexity may be lowered.

Acknowledgement The authors thank the support of Aus-
tralian Research Council (DP120102489 and DP150106118),
National Natural Science Foundation of China (No.61272415
and No.61003056), and Fundamental Research Funds for the
Central Universities of China (No.21615441).

References

[Alur et al., 2002] Rajeev Alur, Thomas A. Henzinger, and
Orra Kupferman. Alternating-Time Temporal Logic.

[Busard et al., 2013] Simon Busard, Charles Pecheur,
Hongyang Qu, and Franco Raimondi. Reasoning about
strategies under partial observability and fairness
constraints. In Ist Workshop on Strategic Reasoning 2013

[Cermák et al., 2014] Petr Cermák, Alessio Lomuscio,
Fabio Mogavero, and Aniello Murano. Mcmas-slk: A
model checker for the verification of strategy logic speci-
fications. In 26th International Conference on Computer


