# 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].

| Logic | Exp. Rep. | Con. Rep. | Sym. Rep. |
|---|---|---|---|
| CTL | PTIME | PSPACE | PSPACE |
| CTL$^*$ | PSPACE | PSPACE | PSPACE |
| ATL$_{ir}$ | $\Delta_2^P$ | PSPACE | NEXPTIME |
| ATL$^*_{ir}$ | PSPACE | PSPACE | NEXPTIME |

Table 1: Model Checking Complexities

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 [Cimatti *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-

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 $\mathtt{Var}$ be a set of atomic propositions. The syntax of the language ATL$^*$ is as follows:

$$\begin{array}{lll} \phi & ::= & p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \langle\!\langle G \rangle\!\rangle \phi \mid E\varphi \\ \varphi & ::= & \phi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2 \end{array}$$

where $p \in \mathtt{Var}$ 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 $\langle\!\langle G \rangle\!\rangle$ 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 $A$ [2]. 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 state 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 $\mathtt{Act} = \Pi_{i \in Agt} \mathtt{Act_i}$ be a set of joint actions, where $\mathtt{Act_i}$ 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 $(S, I, \{N_i\}_{i \in Agt}, \{O_i\}_{i \in Agt}, \longrightarrow, \pi)$ where $S$ is a set of states, $I \subseteq S$ is a set of initial states, $N_i : S \to \mathcal{P}(\mathtt{Act_i}) \setminus \{\emptyset\}$ assigns each state a nonempty set of *legal* actions that may be taken by agent $i$, $O_i : S \to O$ provides agent $i$ with an observation on each state, $\longrightarrow \subseteq S \times \mathtt{Act} \times S$ is a transition relation, and $\pi : S \to \mathcal{P}(\mathtt{Var})$ labels each state with a set of atomic propositions. We assume that the transition relation $\longrightarrow$ is serial, i.e., for every joint action $a \in \mathtt{Act}$ and every state $s$, there exists a state $t$ such that $(s, a, t) \in \longrightarrow$.

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 $\theta_j$ for some $j \in Agt$, we write $M\theta_j$ for the system $(S, I, \{\theta_j\} \cup \{N_i\}_{i \neq j, i \in Agt}, \{O_i\}_{i \in Agt}, \longrightarrow, \pi)$ where the legal action function $N_j$ of agent $j$ is replaced with the strategy $\theta_j$. We say that the agent $j$ follows strategy $\theta_j$ in system $M\theta_j$. 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 $s_0 s_1 ...$, such that $\exists a \in \mathtt{Act} : (s_i, a, s_{i+1}) \in \longrightarrow$ 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 $\phi$.

- $M, s \models p$ for $p \in \mathtt{Var}$ 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 $M, s \models \phi_2$
- $M, s \models \langle\!\langle G \rangle\!\rangle \phi$ 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 $M, \rho[1] \models \varphi$
- $M, \rho \models \varphi_1 U \varphi_2$ 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 $\langle\!\langle G \rangle\!\rangle \phi$, 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 $|\mathtt{Act}|$ of actions, and the size $|\phi|$ of formula. The size of transition relation is polynomial with respect to both $|S|$ and $|\mathtt{Act}|$. 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 $\mathtt{Var}$ of atomic propositions is partitioned into disjoint sets $\mathtt{Var_x}$ for $x \in Agt \cup \{e\}$. Let $\mathtt{Obs_i}$ be the set of observations which agent $i \in Agt$ can observe from environment

[2] The variant of immediately prefixing every path formula with a strategy operator can be expressed with this syntax.

states. The environment $A_e$ is a tuple $(L_e, I_e, \{P_i\}_{i\in Agt}, \longrightarrow_e, \pi_e)$, where $L_e$ is a set of environment states, $I_e \subseteq L_e$ is a set of initial states, $P_i : L_e \to \mathtt{Obs_i}$ provides agent $i$ with an observation on each environment state, $\longrightarrow_e \subseteq L_e \times \mathtt{Act} \times L_e$ is a transition relation, and $\pi_e : L_e \to \mathcal{P}(\mathtt{Var_e})$ is a labelling function. Note that $|\mathtt{Obs_i}| \leq |L_e|$. 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 $i \in Agt$, is a tuple $(L_i, I_i, \longrightarrow_i, \pi_i)$, where $L_i$ is a set of local states, $I_i \subseteq L_i$ is a set of initial states, $\longrightarrow_i \subseteq L_i \times \mathtt{Obs_i} \times \mathtt{Act_i} \times L_i$ is a transition relation: a tuple $(l_i, o_i, a_i, l_i') \in \longrightarrow_i$ means that when 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 $L_i \times \mathtt{Obs_i}$ to $\mathcal{P}(\mathtt{Act_i})$.

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

1. $S = L_e \times \Pi_{i\in Agt} L_i$, $I = I_e \times \Pi_{i\in Agt} I_i$,
2. for all states $s \equiv (l_e, l_1, ..., l_n)$, $O_i(s) = (P_i(l_e), l_i)$,
3. for all states $s \equiv (l_e, l_1, ..., l_n) \in S$ and all $a_i \in \mathtt{Act_i}$, we let $a_i \in N_i(s)$ if and only if there exists a local state $l_i' \in L_i$ such that $(l_i, P_i(l_e), a_i, l_i') \in \longrightarrow_i$,
4. for all states $s \equiv (l_e, l_1, ..., l_n)$, $s' \equiv (l_e', l_1', ..., l_n')$ and joint actions $a \equiv (a_1, ..., a_n)$, we have that $(s, a, s') \in \longrightarrow$ if and only if $(l_e, a, l_e') \in \longrightarrow_e$ and for all agents $i \in Agt$, there is $(l_i, P_i(l_e), a_i, l_i') \in \longrightarrow_i$, and
5. $\pi(s) = \bigcup_{x\in Agt\cup\{e\}} \pi_x(l_x)$.

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 over the number $|\bigcup_{x\in Agt\cup\{e\}} L_x|$ of local states, the number $|\bigcup_{i\in Agt} \mathtt{Act_i}|$ 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 [Lomuscio *et al.*, 2009] and MCK [Gammie 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 $\bigcup_{i\in Agt} \mathtt{Act_i}$ can be regarded as atomic propositions. The environment $\mathtt{Agt_e}$ is a tuple $(\mathtt{Var_e}, \mathtt{Ini_e}, \mathtt{Trn_e})$, where $\mathtt{Var_e}$ is a set of environment variables, formula $\mathtt{Ini_e} \in \mathcal{B}(\mathtt{Var_e})$ represents a set of initial states, formula $\mathtt{Trn_e} \in \mathcal{B}(\mathtt{Var_e} \cup \mathtt{Var_e}' \cup \bigcup_{i\in Agt} \mathtt{Act_i})$ 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 $\mathtt{Agt_i}$ is a tuple $(\mathtt{Var_i}, \mathtt{OVar_i}, \mathtt{Ini_i}, \mathtt{Trn_i})$, for $i \in Agt$, where $\mathtt{Var_i}$ is a set of local variables, $\mathtt{OVar_i} \subseteq \mathtt{Var_e}$ is a set of environment variables that are observable to agent $i$, formula $\mathtt{Ini_i} \in \mathcal{B}(\mathtt{Var_i})$ represents a set of initial states, formula $\mathtt{Trn_i} \in \mathcal{B}(\mathtt{OVar_i} \cup \mathtt{Var_i} \cup \mathtt{Var_i}' \cup \mathtt{Act_i})$ 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 $\mathtt{OVar_i} \cup \mathtt{Var_i} \cup \mathtt{Act_i}$.

Given a symbolic representation $F = \{\mathtt{Agt_i}\}_{i\in Agt\cup\{e\}}$, we can construct its corresponding labeled transition system $M(F) = (S, I, \{N_i\}_{i\in Agt}, \{O_i\}_{i\in Agt}, \longrightarrow, \pi)$ such that

1. $S = sa(\mathtt{Var}, True)$, $I = sa(\mathtt{Var}, \mathtt{Ini_e} \wedge \bigwedge_{i\in Agt} \mathtt{Ini_i})$,
2. for all states $s \in S$ and all $a_i \in \mathtt{Act_i}$, we let $a_i \in N_i(s)$ if and only if $\mathtt{Trn_i} \wedge s \wedge a_i \neq False$ [3],
3. $\longrightarrow = sa(\mathtt{Var} \cup \mathtt{Var}' \cup \bigcup_{i\in Agt} \mathtt{Act_i}, \mathtt{Trn_e} \wedge \bigwedge_{i\in Agt} \mathtt{Trn_i})$,
4. for all states $s \in S$, $O_i(s) = s \upharpoonright (\mathtt{OVar_i} \cup \mathtt{Var_i})$, 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 $|\mathtt{Var}|$ of atomic propositions, the number $|\bigcup_{i\in Agt} \mathtt{Act_i}|$ 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, Esterel and Lustre) which have been widely used in modelling reactive systems. On the other hand, concurrent programs base the idea on process algebraic

[3] A state or an observation can be represented either as a set of literals (variables or their negations), or the conjunction of them.

languages, which have been extensively studied for modelling asynchronous processes.

A concurrent program can be seen as a complete information multiagent 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 $\texttt{Act}_\texttt{i}$ which may be overlapping, and the product system has actions $\bigcup_{i\in Agt}\texttt{Act}_\texttt{i}$. A tuple $(s,a,t)$ is a transition between states $s \equiv (l_1,...,l_n)$ and $t \equiv (l'_1,...,l'_n)$, if for all $i \in Agt$, 1) $a \in \texttt{Act}_\texttt{i}$ implies $(l_i,a,l'_i) \in \longrightarrow_i$, and 2) $a \notin \texttt{Act}_\texttt{i}$ implies $l_i = l'_i$. Due to different constructions, we can not directly derive complexity results of concurrent representation from those of 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 $\text{ATL}_{Ir}$ and $\text{ATL}_{ir}$ model checking, an MIS will be unfolded into different explicit representations. They conjectured that although $\text{ATL}_{Ir}$ model checking is easier than that of $\text{ATL}_{ir}$ in explicit representation, it is harder in MIS. This is different with our results which are based on multiagent systems.

**Theorem 1** *The complexity of model checking CTL is PSPACE-hard for the concurrent representation of multiagent 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 $b$, $\delta : Q\times\Gamma \to \mathcal{P}(Q\times\Gamma\times\{-1,1\})$ 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., $2\times|\Gamma|^2\times|Q|^2$. 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 $Agt = \{1,...,n\}$ such that each tape cell is controlled by an agent. For $i \in Agt$, we define $i\oplus 1 = i+1$, if $i<n$, and $=n$, otherwise. Moreover, $i\oplus -1 = i-1$, if $i>1$, and $=1$, otherwise. Let $\texttt{Act}_\texttt{i} = \{\tau\}\cup\{act_a \mid a\in\Gamma\setminus\{b\}\}$ for $i\in Agt$. We write $a_i$ for agent $i$'s local action in the joint action $a$. The environment $A_e$ is $(L_e,I_e,\{P_i\}_{i\in Agt},\longrightarrow_e,\pi_e)$ 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. $P_i(s) = s$, i.e., agents can see the environment state,
4. for all $(q,a,q',a',d) \in \delta$, we let $((q,m),ja,(q',m\oplus d)) \in \longrightarrow_e$ for all $1\le m\le n$, if $ja_m = act_{a'}$ and $ja_k = \tau$ for all $k\neq m$, and
5. $acc \in \pi_e((q,m))$ for all $1\le m\le n$, if $q\in F$.

Let $\texttt{Obs}_\texttt{i} = L_e$. Agent $A_i$ is $(L_i,I_i,\longrightarrow_i,\pi_i)$ 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 $\longrightarrow_i$ 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 $C = \{A_i\}_{i\in Agt\cup\{e\}}$ simulates the computation of the LBA, we first see that $((q_0,1),b,...,b)$, 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 $M(C)\models EF\,acc$. □

**Theorem 2** *The complexity of model checking $\text{ATL}^*_{ir}$ is in PSPACE for the concurrent representation of multiagent systems.*

**Proof:** We present a PSPACE model checking algorithm for $\text{ATL}^*_{ir}$. 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 $sat(C,s_0,\phi)$.

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

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

The function $psat(C,s,\varphi)$ 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 $A_{\neg\varphi}$ 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 $M(C) \times A_\varphi$ 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 $sat(C, s, \psi)$.

Let $nL = |\bigcup_{x\in Agt\cup\{e\}} L_x|$ and $nA = |\bigcup_{i\in Agt} \mathtt{Act_i}|$. To handle state formulas, the algorithm needs to remember the current state, which takes $O(|Agt|\times\log nL)$ bits, the current strategy $\theta$, which takes up to $\sum_{i\in Agt} |L_i|\times|L_e|\times|\mathtt{Act_i}| = O(|Agt|\times nL^2\times nA)$ bits, and the current formula, which takes up to $O(|\phi|)$ bits of space. To handle path formulas, the algorithm needs up to $O((|Agt| \times \log nL + |\phi|)^2)$ bits of space for the Savitch-style search. Therefore, the space requirement is $sp = O((|Agt| \times \log nL + |\phi|)^2 + |Agt| \times nL^2 \times nA)$.

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

The above theorems lead to the following conclusions.

**Corollary 1** *The complexities of model checking CTL, CTL*, $ATL_{ir}$, $ATL^*_{ir}$ 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 $ATL^*_{ir}$. □

## 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 $ATL_{ir}$ and $ATL^*_{ir}$ 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 $C = \{A_x\}_{x\in Agt\cup\{e\}}$. We introduce two boolean variables $b_s$ and $b'_s$ for each state $s \in \bigcup_{x\in Agt\cup\{e\}} L_x$ and one boolean variable $b_o$ for each observation $o \in \mathtt{Obs_i}$ with $i \in Agt$. We define several formulas:

1. $f_{x,s} = b_s \wedge \bigwedge_{t\in L_x\setminus\{s\}} \neg b_t$, expressing that $s$ is the current state of $x \in Agt \cup \{e\}$, and $f'_{x,s} = b'_s \wedge \bigwedge_{t\in L_x\setminus\{s\}} \neg b'_t$, expressing that $s$ is the next-time state of $x \in Agt \cup \{e\}$,
2. $g_{i,o} = b_o \wedge \bigwedge_{o'\in\mathtt{Obs_i},o'\neq o} \neg b_{o'}$, expressing that the current observation of agent $i$ on the environment state is $o$, and
3. $h_i = \bigvee_{s\in L_e}(f_{e,s} \wedge g_{i,P_i(s)})$, expressing the function $P_i$ by the relation between states and observations.

For a joint action $a \equiv (a_1, ..., a_n)$, we let $k_a = \bigwedge_{i\in Agt} a_i$. We construct $\mathtt{Agt_e} = (\mathtt{Var_e}, \mathtt{Ini_e}, \mathtt{Trn_e})$ such that

1. $\mathtt{Var_e} = \{b_s \mid s \in L_e\} \cup \bigcup_{i\in Agt}\{b_o \mid o \in \mathtt{Obs_i}\}$,
2. $\mathtt{Ini_e} = \bigwedge_{i\in Agt} h_i \wedge \bigvee_{s\in I_e} f_{e,s}$, and
3. $\mathtt{Trn_e} = \bigwedge_{i\in Agt} h_i \wedge \bigvee_{(s,a,t)\in\longrightarrow_e} f_{e,s} \wedge k_a \wedge f'_{e,t}$.

Intuitively, in $\mathtt{Trn_e}$, the formula $\bigvee_{(s,a,t)\in\longrightarrow_e} f_{e,s} \wedge k_a \wedge f'_{e,t}$ encodes all possible transitions, and then the formula $\bigwedge_{i\in Agt} h_i$ tells the observations of agents. It is similar for $\mathtt{Ini_e}$. Moreover, we have $\mathtt{Agt_i} = (\mathtt{Var_i}, \mathtt{OVar_i}, \mathtt{Ini_i}, \mathtt{Trn_i})$ such that

1. $\mathtt{Var_i} = \{b_s \mid s \in L_i\}$, $\mathtt{OVar_i} = \{b_o \mid o \in \mathtt{Obs_i}\}$,
2. $\mathtt{Ini_i} = \bigvee_{s\in I_i} f_{i,s}$ and $\mathtt{Trn_i} = \bigvee_{(s,o,a,t)\in\longrightarrow_i} f_{i,s} \wedge g_{i,o} \wedge a \wedge f'_{i,t}$.

From the way of constructing its explicit representation, a global transition $\longrightarrow$ will need to have the same $g_{i,o}$ on both $\mathtt{Trn_i}$ and $\mathtt{Trn_e}$. 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 $F = \{\mathtt{Agt_i}\}_{i\in Agt\cup\{e\}}$ 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 $M(C) \models \phi$ if and only if $M(F) \models \phi$.

The upper bound can be obtained by reusing the algorithm in Theorem 2. We only describe the differences. First, the procedure for $sat(C, s, \langle\langle G\rangle\rangle\phi)$ 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 $\mathtt{Var}$. Third, the evaluation of transitions between states are done by guessing a joint action and then evaluating the satisfiability of the boolean formula $\mathtt{Trn_e} \wedge \bigwedge_{i\in Agt} \mathtt{Trn_i}$, 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 $ATL_{ir}$ and $ATL^*_{ir}$ are NEXP-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 $X_1, ...X_n, Y_1, ...Y_n$ be tuples of boolean variables and $F(X_1, ..., X_n, Y_1, ..., Y_n)$ be a boolean formula over these variables. A DQBF formula can be written as

$$\forall X_1 ... \forall X_n \exists Y_1(X_1) ... \exists Y_n(X_1, ..., X_n) : \ F(X_1, ..., X_n, Y_1, ..., Y_n).$$

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 $g_1(X_1)$ (in variables $X_1$) through $g_n(X_1, \ldots, X_n)$ (in variables $X_1, \ldots, X_n$) such that the QBF formula

$$\forall X_1 ... \forall X_n(F(X_1, ..., X_n, g_1(X_1), \ldots, g_n(X_1, \ldots, X_n)))$$

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 $X = X_1 \cup ... \cup X_n$ and $Y = Y_1 \cup ... \cup Y_n$. The system consists of a set of agents $Agt = Y$. 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., $\mathtt{Act_y} = \{setT_y, setF_y\}$.

The environment represents the $X$ and $Y$ variables and handles the evaluation of the formula $F$. More specifically, we have $\mathtt{Agt_e} = (\mathtt{Var_e}, \mathtt{Ini_e}, \mathtt{Trn_e})$ such that

1. $\mathtt{Var_e} = X \cup Y \cup \{f\}$, $\mathtt{Ini_e} = True$,
2. $\mathtt{Trn_e} = \bigwedge_{y \in Y}((setT_y \Rightarrow y') \wedge (setF_y \Rightarrow \neg y')) \wedge (f \Leftrightarrow F(X_1, ..., X_n, Y'_1, ..., Y'_n)$,

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 $setT_y$. The value of the formula $F$, assigned to the variable $f$, is then computed by taking the next-time values. For every $k = 1 \ldots n$ and variable $y \in Y_k$, we have agent $\mathtt{Agt_y} = (\mathtt{Var_y}, \mathtt{OVar_y}, \mathtt{Ini_y}, \mathtt{Trn_y})$ such that

1. $\mathtt{Var_y} = \emptyset$, $\mathtt{OVar_y} = X_1 \cup \ldots \cup X_k$ consists of the set of $X$ variables on which $y$ may depend,
2. $\mathtt{Ini_y} = True$, and $\mathtt{Trn_y} = setT_y \vee setF_y$.

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 $M(F) \models \langle\langle Y \rangle\rangle AXf$.

For the upper bound, we can reuse the algorithm in Theorem 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 $\theta(v)$, where the input variables are $\mathtt{OVar_i} \cup \mathtt{Var_i} \cup \mathtt{Act_i}$. Each time when dealing with formula $\langle\langle G \rangle\rangle \phi$, 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 $\varphi$, we explicitly construct the product automaton $M(F) \times A_\varphi$, which is of exponential size with respect to both $F$ and $\varphi$. Note that, we do not explicitly construct $F_0[\theta_G]$. 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$_{ir}$, ATL$^*_{ir}$) on two succinct representations of multiagent systems. For concurrent representation, it is shown that all of them are PSPACE-complete. On the other hand, for symbolic representation, the complexities for branching time logics remain at PSPACE-complete, while they are NEXPTIME-complete for ATL$_{ir}$ and ATL$^*_{ir}$. 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$_{ir}$ and ATL$^*_{ir}$. There are only a few attempts. In [Lomuscio 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 actions 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 logics with richer expressiveness, e.g., [Huang and van der Meyden, 2014c; 2014a; Cermák *et al.*, 2014], or different memory requirements such as perfect recall, where agents have memory 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 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 representations of complete information systems. Complete information systems can be seen as special cases of incomplete information systems, such that agents can observe the underlying system state. It is therefore reasonable to expect that the complexity may be lowered.

**Acknowledgement** The authors thank the support of Australian Research Council (DP120102489 and DP150101618), 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 Orna Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[Busard *et al.*, 2013] Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. Reasoning about strategies under partial observability and fairness constraints. In *1st Workshop on Strategic Reasoning 2013 (SR'13)*, pages 71–79, 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 specifications. In *26th International Conference on Computer Aided Verification (CAV2014)*, pages 525–532, 2014.

[Chandra *et al.*, 1980] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1980.

[Cimatti *et al.*, 2002] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *14th International Conference on Computer Aided Verification (CAV2002)*, pages 359–364, 2002.

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

[Clarke *et al.*, 1999] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[Emerson and Lei, 1987] E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

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

[Guelev *et al.*, 2011] Dimitar P. Guelev, Catalin Dima, and Constantin Enea. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.

[Huang and van der Meyden, 2010] Xiaowei Huang and Ron van der Meyden. The complexity of epistemic model checking: Clock semantics and branching time. In *19th European Conference on Artificial Intelligence (ECAI2010)*, pages 549–554, 2010.

[Huang and van der Meyden, 2014a] Xiaowei Huang and Ron van der Meyden. An epistemic strategy logic. In *the 2nd International Workshop on Strategic Reasoning (SR2014)*, pages 35–41, 2014.

[Huang and van der Meyden, 2014b] Xiaowei Huang and Ron van der Meyden. Symbolic model checking epistemic strategy logic. In *Proceedings of the the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, 2014.

[Huang and van der Meyden, 2014c] Xiaowei Huang and Ron van der Meyden. A temporal logic of strategic knowledge. In *the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR2014)*, 2014.

[Huang, 2015] Xiaowei Huang. Bounded model checking of strategy ability with perfect recall. *Artificial Intelligence*, pages 182–200, 2015.

[Jamroga and Agotnes, 2007] Wojciech Jamroga and Thomas Agotnes. Modular interpreted systems. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, page 131, 2007.

[Jamroga and Dix, 2008] Wojciech Jamroga and Jurgen Dix. Model checking abilities of agents: A closer look. *Theory of Computing Systems*, 42(3):366–410, 2008.

[Kacprzak *et al.*, 2008] Magdalena Kacprzak, Wojciech Nabiałek, Artur Niewiadomski, Wojciech Penczek, Agata Półrola, Maciej Szreter, Bożena Woźna, and Andrzej Zbrzezny. VerICS 2007 - a Model Checker for Knowledge and Real-Time. *Fundamenta Informaticae*, 85(1):313–328, 2008.

[Kupferman *et al.*, 2000] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *J. ACM*, 47(2):312–360, 2000.

[Lomuscio and Raimondi, 2006] Alessio Lomuscio and Franco Raimondi. 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)*, pages 161–168, 2006.

[Lomuscio *et al.*, 2009] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *Proc. Conf. on Computer-Aided Verification*, pages 682–688, 2009.

[Peterson *et al.*, 2001] Gary Peterson, John Reif, and Salman Azhar. Lower bounds for multiplayer non-cooperative games of incomplete information. *Computers and Mathematics with Applications*, 41:957–992, 2001.

[Pnueli, 1977] Amir Pnueli. The Temporal Logic of Programs. In *Symp. on Foundations of Computer Science*, pages 46–57, 1977.

[Savitch, 1970] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

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

[van der Meyden and Shilov, 1999] Ron van der Meyden and Nikolay V. Shilov. Model Checking Knowledge and Time in Systems with Perfect Recall. In *Foundations of Software Technology and Theoretical Computer Science*, pages 432–445, 1999.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. Automata theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32(2):183–221, 1986.