Bounded Planning for Strategic Goals with Incomplete Information and Perfect Recall*

Xiaowei Huang
School of Computer Science and Engineering
University of New South Wales, Australia
xiaoweih@cse.unsw.edu.au

ABSTRACT

The paper proposes an OBDD-based bounded model checking algorithm for alternating-time temporal logic in systems of incomplete information and multiple players. Players are assumed to have perfect recall memory over their observations and local actions. The algorithm is implemented in a model checker and experimental results are reported to show its applications in bounded planning for strategic goals. The computational complexity of model checking is also addressed.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence/Automatic Programming**]: Program verification

Keywords

Strategic Logic, Multi-agent Systems, Model Checking

1. INTRODUCTION

Planning is a key capability of intelligent systems and has been an area of research in artificial intelligence for over three decades. Planning increases the autonomy of an intelligent system through the construction of plans to achieve *goals*. Planning techniques have been applied in various applications including robotics, autonomous agents, web-based information gathering, and spacecraft mission control.

Planning has a strong interaction with logic-based knowledge representation and reasoning schemes, including the representation of world models, reasoning about the effects of actions, and techniques for efficiently searching the space of possible plans by exploiting logical structure of problems. Given a start state, a goal condition, and available actions, the objective of planning is to find a sequence of actions leading from start to goal. Most planning techniques are devoted to design specialized algorithms to deal with specific systems and goals.

Planning as model checking has been shown to be an effective technique for planning, and is fully motivated by the flexibility of describing goals as logic formulas and the fast development of the area of model checking. It first works with temporal logics [12] and then extends to the logic of knowledge [19] and ATL [13, 20]. Currently, using ATL model checking for planning focuses on memoryless strategies or complete information systems.

Alternating-time temporal logic (ATL) [2] has attracted many researches and applications by its ability in reasoning about systems of multiple players. In a multiplayer system, players make moves by following strategies (or plans). A memoryless strategy decides the next action by utilising the information available in the current round. A perfect recall strategy decides the next action by utilising all available information up to the current round.

In many practical systems like poker games, players are not supposed to observe everything, especially not able to observe the local states of their opponents. A perfect recall player can make maximal use of reasoning capabilities. Therefore, perfect recall is an optimal assumption over the adversary and a meaningful setting when designing and verifying critical systems.

The strategic aspect of the ATL is its capability to express, as a formula $\langle\langle A\rangle\rangle\phi$, the semantics of "a set of players have a strategy to enforce ϕ". The incomplete information may result in several different interpretations on the statement that a set of players have a strategy, including the existence of a strategy, the existence of a consistent strategy, knows the existence of a consistent strategy but don't know how to play, and knows not only the existence of a strategy but also how to play, see e.g., [21]. In the paper, we assume the last one which is the most suitable one for planning.

Model checking ATL of incomplete information and perfect recall is believed to be undecidable [2]. Although this pessimistic conjecture prevents us from working with the full logic, we might still work with its decidable fragments. In this paper, we propose a model checking algorithm to enable the reasoning in a bounded number of steps.

The idea of bounded model checking [6] has achieved success in software verification, by reducing the model checking problem to the satisfiability problem. Starting from temporal logics, it has been extended to deal with the logic of knowledge, in e.g., [25, 15]. However, bounded model checking technique based on SAT solvers usually deals with the positive fragment of a logic to avoid the alternations between existential and universal quantifications. Therefore, it can not be directly extended to ATL logic, as the semantics of each coalition operator needs a nesting of existential and universal quantifications.

This paper makes the following contributions: Firstly, the semantics of ATL is settled on a variant of interpreted systems [10] and the partially-observed concurrent game structures. Secondly, we show that the computational complexity of model checking ATL formulas of bounded depth is PSPACE-complete. Thirdly, a symbolic model checking algorithm based on OBDDs is proposed to deal with ATL formulas of bounded depth and implemented in the

*Research supported by Australian Research Council Discovery Grants DP1097203 and DP120102489.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013),* Ito, Jonker, Gini, and Shehory (eds.), May 6–10, 2013, Saint Paul, Minnesota, USA.
Copyright © 2013, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

(Carl, Adam)	cooperate	defect
cooperate	(3,3)	(0,5)
defect	(5,0)	(1,1)

Table 1: Payoff of Prisoner's Dilemma

epistemic model checker MCK [11]. This is the first time a practical algorithm and tool is brought forward for ATL with incomplete information and synchronous perfect recall. We prove its correctness. Finally, the tool is applied to a set of applications, including the iterated prisoner's dilemma, Kriegspiel tic-tac-toe, and a patrolling game.

2. RUNNING EXAMPLE: ITERATED PRISONER'S DILEMMA

Before proceeding to the technical details, let's have a look at the *Iterated Prisoner's Dilemma* (IPD) [4], which is a canonical game in game theory, and has been experimentally analysed by tournaments held in 1980 [3], in which contestants submitted strategies for a 200-repetition Prisoner's Dilemma, and the strategies could not be updated during play. The tournament was held again in 2004 and 2005.

Prisoner's Dilemma (PD): Two men `Carl` and `Adam` are arrested, but the police do not possess enough information for a conviction. Following the separation of the two men, the police offer both a similar deal: if one testifies against his partner (`defect`) and the other remains silent (`cooperate`), then the betrayer goes free and the one that remains silent receives a 5-month sentence. If both remain silent, both are sentenced to only one month in jail for a minor charge. If each 'rats out' the other, each receives a three-month sentence. Each prisoner must choose either to betray or remain silent.

Instead of using penalties as described above, we use rewards to ease the following interpretation. Table 1 gives the rewards. That is, $\texttt{rwd}(a_1, a_2)$ denotes the rewards of the pair of actions (a_1, a_2), taken by `Carl` and `Adam`, respectively. For example, we have $\texttt{rwd}(\texttt{cooperate}, \texttt{defect}) = (0, 5)$, which says that if `Carl` chooses to cooperate and `Adam` chooses to defect, then the former gets 0 rewards and the later gets 5 rewards. An IPD game is a repeated PD game, such that two players play PD more than once in succession and they remember previous actions of their opponent and may change their actions accordingly.

In the games that we are concerned, the player `Adam` is given a set of strategies (details will be given later), from which he can choose one to follow. The player `Carl` is not informed about which strategy his opponent is following. The incomplete information exists because `Carl` is not sure about `Adam`'s strategy. The perfect recall is useful for `Carl` as he may use it to reason, from the past observations, about the strategy `Adam` is following.

The goal of the player `Carl` is to maximise the benefits (expressed in terms of logic formulas) under the incomplete information. The game can be seen as a single-player game of the player `Carl`, because the other player `Adam` will follow a specific strategy that is chosen initially.

3. BOUNDED ATL LOGIC

Suppose that we are working with a system of a finite set $Agt = \{1, \ldots, n\}$ of players. Let $Prop$ be a set of propositions. The logic ATL [2] combines the temporal operators and the strategic operator to reason about the strategic ability of the players. The bounded ATL logic has the syntax of

$$\begin{aligned}\phi \quad ::= \quad & p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \\ & \langle\!\langle A \rangle\!\rangle X\phi \mid \langle\!\langle A \rangle\!\rangle \phi_1 \, U^{\leq k} \phi_2 \mid \langle\!\langle A \rangle\!\rangle \phi_1 \, R^{\leq k} \phi_2\end{aligned}$$

where $p \in Prop$, $k \geq 0$, and $A \subseteq Agt$. Instead of setting a specific bound to terminate model checking whenever the bound is reached, as is done in traditional bounded model checking, we impose bounds on formulas to pursue greater flexibility in specifying properties.

Intuitively, formula $\langle\!\langle A \rangle\!\rangle X\phi$ expresses that players in A can collaboratively enforce the fact ϕ at the next time, $\langle\!\langle A \rangle\!\rangle \phi_1 \, U^{\leq k} \phi_2$ means that, within k steps, the players in A have a strategy to keep enforcing ϕ_1 until ϕ_2 holds, and $\langle\!\langle A \rangle\!\rangle^{\leq k} \phi_1 \, R \, \phi_2$ means that, within k steps, players in A have a strategy to make sure that only when ϕ_1 holds can ϕ_2 be released. Other operators can be obtained in the usual way, e.g., $\langle\!\langle A \rangle\!\rangle F^{\leq k} \phi \equiv \langle\!\langle A \rangle\!\rangle True \, U^{\leq k} \phi$, $\langle\!\langle A \rangle\!\rangle G^{\leq k} \phi \equiv \langle\!\langle A \rangle\!\rangle False \, R^{\leq k} \phi$, etc.

For instance, in the IPD example, let *carlRwd* be the accumulated rewards of player `Carl` and *adamRwd* be the accumulated rewards of player `Adam`, we may want to check the following specification

$$\langle\!\langle \{\texttt{Carl}\} \rangle\!\rangle (True \, U^{\leq k} (carlRwd > adamRwd)) \tag{1}$$

which expresses that the player `Carl` has a strategy to achieve, within k iterations, a better rewards than its opponent `Adam`. The second specification

$$(\langle\!\langle \{\texttt{Carl}\} \rangle\!\rangle X)^k (carlRwd > k) \tag{2}$$

expresses that the player `Carl` can achieve, after k iterations, a rewards more than that defined by Nash Equilibrium, i.e., both of them always `defect`. The third query we might be interested is, if the player `Carl` has the strategy to achieve, after k iterations, the best combined scores, i.e., both of them always `cooperate`,

$$(\langle\!\langle \{\texttt{Carl}\} \rangle\!\rangle X)^k (carlRwd + adamRwd \geq 6k) \tag{3}$$

4. INTERPRETED SYSTEMS SEMANTICS

We enrich an interpreted system [10] by actions performed by the players, and call the resulting system an action interpreted system (AIS). At all times in an AIS, each player is assumed to be in some *local* state that records all the information that the player can access at that time. The environment e records "everything else that is relevant".

Let S be the set of environment states and let L_i be the set of local states of player $i \in Agt$. A *global* state s of a multi-player system is an $(n+1)$-tuple $(s_e, s_1, \ldots, s_n)$ such that $s_e \in S$ and $s_i \in L_i$ for all $i \in Agt$. At a global state, each player independently takes some local action, which represents the decision it makes. In the meantime, the environment takes an action to update its state. Let Act_e be the set of environment actions and Act_i be the set of local actions of player $i \in Agt$. A *global* action of a multi-player system in some global state is a $(n+1)$-tuple $a = (a_e, a_1, \ldots, a_n)$ such that $a_e \in Act_e$ and $a_i \in Act_i$ for all $i \in Agt$. Let $Act = Act_e \times Act_1 \times ... \times Act_n$.

Time is represented discretely by using natural numbers. A *run* is a function $r : \mathbf{N} \to S \times L_1 \times \ldots \times L_n \times Act_e \times Act_1 \times \ldots \times Act_n$ from time to global states and actions. A pair (r, m) consisting of a run r and time m is called a *point*, which may also be written as $r(m)$. If $r(m) = (s_e, s_1, \ldots, s_n, a_e, a_1, \ldots, a_n)$ then we define $s_e(r, m) = s_e$, $a_e(r, m) = a_e$ and $s_i(r, m) = s_i$ and $a_i(r, m) = a_i$ for $i \in Agt$. If r is a run and m a time, we write $s_e(r, 0..m)$ for the sequence $s_e(r, 0) \ldots s_e(r, m)$, and $a(r, 0..m)$ for $a(r, 0) \ldots a(r, m)$.

Let a system $\mathcal{R}$ be a set of runs, and we call $\mathcal{R} \times \mathbf{N}$ the *set of points* of $\mathcal{R}$. Relative to a system $\mathcal{R}$, we define the set $\mathcal{K}_i(r, m) =$

$\{(r', m') \in \mathcal{R} \times \mathbf{N} \mid view_i(r', m') = view_i(r, m)\}$ to be the set of points that are, for player i, indistinguishable from the point (r, m). The view function $view_i$ will be defined later.

For a system $\mathcal{R}$ of runs, we define a *cell* c to be a set of runs $\mathcal{R}_c$ such that $\mathcal{R}_c \subseteq \mathcal{R}$. (In the game structure semantics presented in the following section, $\mathcal{R}_c$ will be made concrete as the set of runs *compatible* with the strategies that define c.) A point (r, m) is in c if $r \in \mathcal{R}_c$. The set of indistinguishable points for player i in (r, m) assuming c is $\mathcal{K}_i^c(r, m) = \mathcal{K}_i(r, m) \cap \{(r, m) \mid r \in \mathcal{R}_c, m \in \mathbf{N}\}$.

Two cells c_1 and c_2 are *strategic equivalent* for player i, denoted as $c_1 \simeq_i c_2$, if for any two points $(r, m), (r', m')$ in c_1 or c_2, $view_i(r, m) = view_i(r', m')$ implies $a_i(r, m) = a_i(r', m')$. Intuitively, we assume *deterministic strategy* for all players. Note that, the relation $\simeq_i$ is an equivalence relation, i.e., it is reflexive, symmetric, and transitive. We use $[c]_C^{\simeq_i}$ to denote the equivalence class of c in C with respect to the relation $\simeq_i$ and $[\simeq_i]_C$ to denote the set of all equivalence classes in C with respect to the relation $\simeq_i$.

An *action interpreted system* (AIS) is a tuple $(\mathcal{R}, C, \{\simeq_i\}_{i \in Agt}, \pi)$, where $\mathcal{R}$ is a system of runs, C is a set of cells in $\mathcal{R}$ such that $\mathcal{R} = \bigcup\{\mathcal{R}_c \mid c \in C\}$, $\{\simeq_i\}_{i \in Agt}$ is a set of strategic equivalences over cells for all players in Agt, and $\pi : \mathcal{R} \times \mathbb{N} \to \mathcal{P}(Prop)$ is an interpretation such that $\pi(r, m) = \pi(s_e(r, m))$ for all points (r, m).

Let $A \subseteq Agt$ be a set of players. We assume the distributed knowledge among them, that is, define $\mathcal{K}_A(r, m) = \bigcap_{i \in A} \mathcal{K}_i(r, m)$, $\mathcal{K}_A^c(r, m) = \bigcap_{i \in A} \mathcal{K}_i^c(r, m)$. Moreover, we let $\simeq_A = \bigcap_{i \in A} \simeq_i$ and $a_A(r, m) = \{a_i(r, m) \mid i \in A\}$ be the collective action of players in A at point (r, m). Likewise, we can define $[c]_C^{\simeq_A}$ and $[\simeq_A]_C$.

DEFINITION 1. *The semantics of the language in an AIS $\mathcal{I} = (\mathcal{R}, C, \{\simeq_i\}_{i \in Agt}, \pi)$ is given by interpreting formulas ϕ at points (r, m) of $\mathcal{I}$, using a satisfaction relation $\mathcal{I}, (r, m) \models \phi$, which is defined inductively as follows.*

- $\mathcal{I}, (r, m) \models p$ *if* $p \in \pi(r, m)$,
- $\mathcal{I}, (r, m) \models \neg\phi$ *if not* $\mathcal{I}, (r, m) \models \phi$
- $\mathcal{I}, (r, m) \models \phi \wedge \psi$ *if* $\mathcal{I}, (r, m) \models \phi$ *and* $\mathcal{I}, (r, m) \models \psi$
- $\mathcal{I}, (r, m) \models \langle\langle A \rangle\rangle X\phi$ *if there exists an equivalence class* $[c]_C^{\simeq_A} \in [\simeq_A]_C$ *of players A such that*
  - *there exists* $r' \in [c]_C^{\simeq_A}$ *and* $m' \in \mathbf{N}$ *such that* $(r', m') \in \mathcal{K}_A(r, m)$ *and*
  - *for all* $r' \in [c]_C^{\simeq_A}$ *and* $m' \in \mathbf{N}$, *if* $(r', m') \in \mathcal{K}_A(r, m)$ *then* $\mathcal{I}, (r', m' + 1) \models \phi$.
- $\mathcal{I}, (r, m) \models \langle\langle A \rangle\rangle \phi \, U^{\leq k} \psi$ *if*
  - $\mathcal{I}, (r, m) \models \psi$ *or* $\mathcal{I}, (r, m) \models \phi \wedge \langle\langle A \rangle\rangle X(\langle\langle A \rangle\rangle \phi \, U^{\leq k-1} \psi)$ *for* $k > 0$, *and*
  - $\mathcal{I}, (r, m) \models \psi$ *for* $k = 0$.
- $\mathcal{I}, (r, m) \models \langle\langle A \rangle\rangle \phi \, R^{\leq k} \psi$ *if*
  - $\mathcal{I}, (r, m) \models \psi$ *and* $\mathcal{I}, (r, m) \models \phi \vee \langle\langle A \rangle\rangle X(\langle\langle A \rangle\rangle \phi \, R^{\leq k-1} \psi)$ *for* $k > 0$, *and*
  - $\mathcal{I}, (r, m) \models \psi \wedge \phi$ *for* $k = 0$.

Intuitively, in the semantics of $\langle\langle A \rangle\rangle X\phi$, the equivalence class $[c]_C^{\simeq_A}$ represents a joint winning strategy of A such that for all joint opponent strategies, $[c]_C^{\simeq_A}$ enforces a *win* on every compatible state. In other word, the players A knows not only the existence of a strategy but also *how to play*. More concretely, the first condition says that the players A have a strategy that is compatible with the current view. The second condition says that under the new strategy, the formula ϕ can be enforced at the next time.

4.0.1 Synchronous Perfect Recall View

Now we define a view function for players. A player i has *synchronous perfect recall* view, denoted as `spr`, in system $\mathcal{R}$ if there exists a set O (of *observations*) such that for each point (r, m) of $\mathcal{R}$, the view $view_i(r, m)$ is a sequence of exactly $(m + 1)$ elements of O and m elements of Act_i. The observation of the player i at a specific point is given by a function $O_i : \mathcal{R} \times \mathbf{N} \to O$. Then the view of player i is defined by

- $view_i(r, 0) = O_i(r, 0)$, and
- $view_i(r, m + 1) = view_i(r, m) \cdot a_i(r, m + 1) \cdot O_i(r, m + 1)$ for all $m \in \mathbf{N}$, where a is some action in Act.

There exists other view functions, e.g., the observational view that $view_i(r, m) = O_i(r, m)$, representing that the player i can only observe the current observation, and the clock view $view_i(r, m) = (m, O_i(r, m))$, representing that the player i can observe the current time and the current observation. In this paper, we focus on the synchronous perfect recall view.

5. GAME STRUCTURE SEMANTICS

In this section we present a finite model called *partially observed concurrent game structure* (PO-CGS) and define a translation to AIS. A finite PO-CGS for a set Agt of players is a tuple $M = (S, Act_e, Act_1, ..., Act_n, N_e, N_1, ..., N_n, O_1, ..., O_n, I, T, \pi)$, where S is a finite set of states, Act_e is the set of actions of the environment e, Act_i is the set of local actions of player $i \in Agt$, $N_i : S \to \mathcal{P}(Act_i)$ indicates the set of actions that are available to player i at a specific state, $I \subseteq S$ is a set of initial states, $T : S \times Act \to S$ is a transition relation, $O_i : S \to O$ is an observation function for player $i \in Agt$, and $\pi : S \to \mathcal{P}(Prop)$ is an interpretation of the atomic propositions $Prop$ at the states. For consistency, we further require that for all states $s_1, s_2 \in S$ and $i \in Agt$, $O_i(s_1) = O_i(s_2)$ implies $N_i(s_1) = N_i(s_2)$.

We treat the set of states S as the states of the environment rather than as the set of global states, and player i's local states are derived from the observation function O_i and the actions in Act_i that i performs. We write $k_i(s) = \{s' \in S \mid O_i(s') = O_i(s)\}$ for the set of states that are observationally indistinguishable to player i from state s.

Executions to Runs

Let $s, s' \in S$ and $a \in Act$. A *path* ρ from a state s is a finite or infinite sequence of states and actions $s_0 a_1 s_1 a_2 s_2 \ldots$ such that $s_0 = s$ and $s_{k+1} \in T(s_k, a_{k+1})$ for all k such that $k < |\rho| - 1$, where $|\rho|$ is the total number of states on ρ. Given a path ρ, we use $s(\rho, m)$ to denote its $(m+1)$-th state, $a(\rho, m)$ to denote its m-th action, in which $a_e(\rho, m)$ is its m-th environment action and $a_i(\rho, m)$ is its m-th local action of agent i. A *fullpath* from a state s is an infinite path from s. A path ρ is *initialized* if $s(\rho, 0) \in I$.

From each initialized fullpath ρ, one may define a run in an AIS satisfying `spr` for all players. Recall that we interpret the states of the PO-CGS as states of the environment, and the global actions of the PO-CGS as actions of the players as well as the environment. Given an initialized fullpath ρ, we obtain a run $\rho^{\mathtt{spr}}$ by defining each point $(\rho^{\mathtt{spr}}, m)$ with $m \in \mathbf{N}$ as follows. The environment state at time $m \geq 0$ is $s_e(\rho^{\mathtt{spr}}, m) = s(\rho, m)$. The environment action and local action are $a_e(\rho^{\mathtt{spr}}, m) = a_e(\rho, m)$ and $a_i(\rho^{\mathtt{spr}}, m) = a_i(\rho, m)$ for time $m \geq 1$, and $a_e(\rho^{\mathtt{spr}}, 0) = a_i(\rho^{\mathtt{spr}}, 0) = \perp$. The view of player i at time m is $view_i(\rho^{\mathtt{spr}}, m) = \perp \cdot O_i(s(\rho, 0)) \cdot a_i(\rho, 1) \cdot \ldots \cdot O_i(s(\rho, m))$, representing that the player remembers all its observations and past local actions, according to `spr`.

Complete Coalition Strategies to Cells

A strategy σ_i of a player i is a function that maps each finite path $\rho = s_0 a_1 s_1 a_2 \dots s_n$ to an action in $N_i(s_n)$. A (finite or infinite) path ρ is compatible with σ_i if $a_{k+1}(i) = \sigma_i(s_0 a_1 \dots s_k)$ for all $k < |\rho|$ where $|\rho|$ is the number of transitions in ρ. Given a PO-CGS M and a strategy σ_i of player i, write $Path(M, \sigma_i)$ for the set of infinite paths in M that are compatible with σ_i. A strategy σ_i is *uniform* if for all paths $\rho, \rho' \in Path(M, \sigma_i)$ and $m \in \mathbf{N}$, we have $view_i(\rho, m) = view_i(\rho', m)$ implies $a_i(\rho, m+1) = a_i(\rho', m+1)$, i.e., i's reactions following σ_i respect its views.

Let A be a set of players. A coalition strategy σ_A fixes a strategy σ_i for each player $i \in A$. We call σ_A a complete coalition strategy if $A = Agt$, or an incomplete coalition strategy if $A \subset Agt$. Given a complete coalition strategy $\sigma_{Agt} = \{\sigma_i \mid i \in Agt\}$, we define a cell c, and obtain a subset of runs $\mathcal{R}_c = \bigcap_{i \in Agt} Path(M, \sigma_i)$. Note that the strategies of players in $Agt \setminus A$ are not required to be uniform, so that they are allowed to perform arbitrary behaviours.

Incomplete Coalition Strategies to Equivalence Classes over Cells

Let $\bar{A} = Agt \setminus A$ be the complement set of players of A. For each incomplete coalition strategy σ_A, there may exist more than one incomplete coalition strategy $\sigma_{\bar{A}}$. As a complete coalition strategy $\sigma_A \cup \sigma_{\bar{A}}$ restricts the system $\mathcal{R}$ into a cell, an incomplete coalition strategy σ_A restricts $\mathcal{R}$ into a set of cells, each of which corresponds with an incomplete coalition strategy of $\sigma_{\bar{A}}$ of players $\bar{A}$. The following statement ascertains that these cells are strategic equivalent.

PROPOSITION 1. *Let σ_A be an incomplete rational strategy of A and $\sigma^1_{\bar{A}}$ and $\sigma^2_{\bar{A}}$ be two incomplete strategies of $\bar{A}$. Let c_1 and c_2 be the cells for complete strategy $\sigma_A \cup \sigma^1_{\bar{A}}$ and $\sigma_A \cup \sigma^2_{\bar{A}}$ respectively. Then we have $c_1 \simeq_A c_2$.*

Here we remark that, a single run $r \in \mathcal{R}$ may belong to different cells or even different equivalence classes. Also, there might exist more than one strategy of coalition A that are mapped to the same equivalence class over cells. Plainly, such strategies may disagree only on incompatible runs.

PO-CGS to AIS

The system M gives us an interpretation π on its states, and we may lift this to an interpretation on the points (r, m) of $\mathcal{R}$ by defining $\pi(r, m) = \pi(s_e(r, m))$. Using the construction above, we then obtain the action interpreted system $\mathcal{I}(M) = \mathcal{I}(\mathcal{R}, C, \{\simeq_i\}_{i \in Agt}, \pi)$ where C is the set of all cells corresponding to complete coalition strategies and $\simeq_i$ is a strategic equivalence of player i over C. We will be interested in the problem of model checking formulas in this system.

A formula ϕ is said to hold in M, written $M \models \phi$, if $\mathcal{I}(M), (r, 0) \models \phi$ for all $r \in \mathcal{R}$. The model checking problem is then to determine, given a PO-CGS M and a formula ϕ, whether $M \models \phi$.

6. ANALYZING ITERATED PRISONER'S DILEMMA

In the IPD games, a game state is a tuple $(carlRwd, adamRwd, \tau)$, where τ is the strategy selected by player `Adam`. Recall that *carlRwd* and *adamRwd* are accumulated rewards of `Carl` and `Adam` and the player `Adam` follows a strategy τ chosen from a set of strategies. Let Φ be the set of strategies provided to player `Adam`. The initial game states are $I = \{(0, 0, \tau) \mid \tau \in \Phi\}$. Intuitively, `Adam`'s strategy is chosen initially.

A play ρ of the game consists of a sequence of finite or infinite number of states and actions $s_0 a_1 s_1 ...$, where $s_0 \in I$, $a_m = (a_{m,\texttt{Carl}}, a_{m,\texttt{Adam}})$ is the global action of the players such that $a_{m,i} \in \{\texttt{cooperate}, \texttt{defect}\}$, and if $s_m = (rw_1, rw_2, \tau)$ and $s_{m+1} = (rw'_1, rw'_2, \tau')$ then $\tau = \tau'$ and $(rw'_1 - rw_1, rw'_2 - rw_2) = \texttt{rwd}(a_{m+1})$ is the reward pair of joint action a_{m+1} in Table 1.

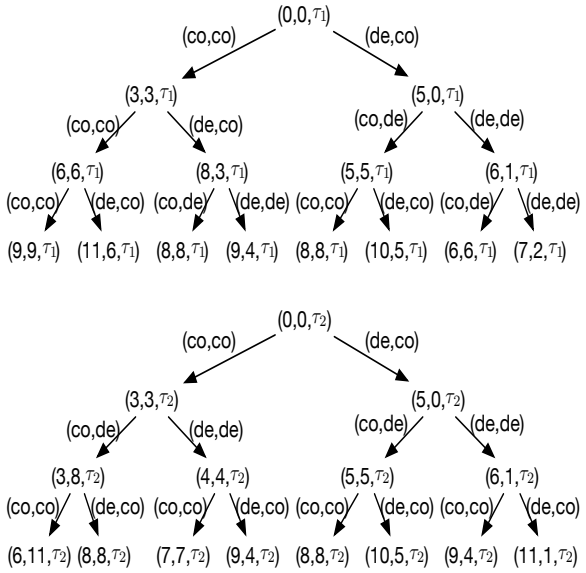

Figure 1: Finite Model of IPD(3)

In each game state, each player makes an observation of the game state. This is captured by an observation function O_i with domain the set of game states for $i \in \{\texttt{Carl}, \texttt{Adam}\}$. We define $O_{\texttt{Carl}}(rw_1, rw_2, \tau) = (rw_1, rw_2)$, which denotes that player `Carl` can not observe the strategy selected by its opponent, and $O_{\texttt{Adam}}(rw_1, rw_2, \tau) = (rw_1, rw_2, \tau)$, which denotes that player `Adam` can observe the game state. A player's perfect recall view of a play $s_0 a_1 s_1 ...$, denoted as $view_i(s_0 a_1 s_1 ...)$, is the sequence of observations and local actions $O_i(s_0) a_{1,i} O_i(s_1) ...$

Recall that, a strategy σ_i for player i maps each finite possible view $\alpha = O_i(s_0) a_{1,i} ... O_i(s_m)$ of the player to a set of actions $\{\texttt{cooperate}, \texttt{defect}\}$. A complete strategy is a mapping σ associating a strategy σ_i to each player i.

Now, in a concrete game, assuming that `Adam`'s strategy can be either

1. τ_1: `cooperate` first and then follow the previous action of `Carl`, or
2. τ_2: alternate between `cooperate` and `defect`.

We use IPD(k) to denote such a game of up to k iterations. Figure 1 gives the finite model for the game IPD(3). Each node (rw_1, rw_2, τ) denotes a game state. An arrow from a node s to another node t denotes the transition relation between the two states. The label (a_1, a_2) on an arrow denotes the action taken by `Carl` and `Adam`, respectively. For abbreviation, in the figure, we write co for `cooperate` and de for `defect`.

First, let's see the capability of synchronous perfect recall in reasoning about strategies. For example, in game states $(8, 8, \tau_1)$ or $(8, 8, \tau_2)$, if the player `Carl` can only observe his current state, then he can't distinguish them by the definition of function $O_{\texttt{Carl}}$. However, with `spr`, if the observation history is $(0, 0)$co$(3, 3)$co$(3, 8)$de$(8, 8)$ then `Carl` knows exactly that `Adam` is following the strategy τ_2. After identifying the opponent's strategy, he can play optimally.

Now, let's find Carl's strategies for the specifications presented before. For specification (1), there exists at least a strategy $\sigma^1_{\texttt{Carl}}$ such that

- $\sigma^1_{\texttt{Carl}}((0,0)) = \texttt{defect}$,
- $\sigma^1_{\texttt{Carl}}((0,0)\text{de}(5,0)) = \texttt{cooperate}$,
- $\sigma^1_{\texttt{Carl}}((0,0)\text{de}(5,0)\text{co}(5,5)) = \texttt{defect}$.

For specification (2), it's not hard to see that Carl can take any strategy.

For specification (3), we notice that there is a path

$$(0,0,\tau_1)(\text{co},\text{co})(3,3,\tau_1)(\text{co},\text{co})(6,6,\tau_1)(\text{co},\text{co})(9,9,\tau_1)$$

leading from an initial state to a state satisfying the proposition $carlRwd + adamRwd \geq 18$. However, the player Carl does not have a strategy to guarantee the achievement of this goal.

7. MODEL CHECKING COMPLEXITY

In this section, we confirm the intuition about the decidability of model checking bounded ATL.

THEOREM 1. *Let M be a PO-CGS of n players, ϕ be a bounded ATL formula, and $k \geq 0$ be the maximal number appears in ϕ. Then the complexity of deciding if $M \models \phi$ is PSPACE-complete.*

PROOF. (sketch) We present an algorithm, different with the one in the next section, for model checking $M \models \phi$. It works on a set of initialized paths of length $|\phi| \cdot k$, i.e.,

$$R_h = \{s_e(r,0)a(r,1)...s_e(r,|\phi|\cdot k) \mid r \in \mathcal{R}\}.$$

Note that the set R_h is of size $O((|M| \cdot |Act|)^{|\phi|\cdot k})$ by letting $|M|$ be the number of states and $|Act|$ be the number of global actions. Our algorithm avoids the explicit construction of this set.

The satisfiability of an expression $R_h, (r,k), \{v_1,...,v_n\} \models \phi$ is computed recursively by the following procedure, where (r,k) is a point in R_h and v_i is a sequence of observations and actions of player i. Intuitively, this expression states that the formula ϕ holds in the point (r,k) of R_h under the observation history v_i for player i.

- Let $\phi = p$. Then $R_h, (r,k), \{v_1,...,v_n\} \models \phi$ if $p \in \pi(r,k)$
- Let $\phi = \neg\phi'$. Then $R_h, (r,k), \{v_1,...,v_n\} \models \phi$ if not $R_h, (r,k), \{v_1,...,v_n\} \models \phi'$
- Let $\phi = \phi_1 \wedge \phi_2$. Then $R_h, (r,k), \{v_1,...,v_n\} \models \phi$ if $R_h, (r,k), \{v_1,...,v_n\} \models \phi_1$ and $R_h, (r,k), \{v_1,...,v_n\} \models \phi_2$
- Let $\phi = \langle\langle A\rangle\rangle X\phi'$. Then $R_h, (r,k), \{v_1,...,v_n\} \models \phi$ if we can
  - existentially choose an equivalence class $[c]_C^{\simeq_A} \in [\simeq_A]_C$, in which there exists at least a run r' such that $r'_i(k) = v_i$ for all $i \in A$, and then
  - universally verify $R_h, (r', k+1), \{v'_1,...,v'_n\} \models \phi'$ for all runs r' such that $r' \in \mathcal{R}_{c'}$, $c' \in [c]_C^{\simeq_A}$ and $r'_i(k) = v_i$ for all $i \in A$, where $v'_i = v_i \cdot a_i(r,k+1) \cdot O_i(r,k+1)$ for $i \in Agt$.
- The cases of $\phi = \langle\langle A\rangle\rangle\phi_1 \, U^{\leq k}\phi_2$ and $\phi = \langle\langle A\rangle\rangle\phi_1 \, R^{\leq k}\phi_2$ are done by unfolding the formulas as the way in the semantics.

Now, to verify $M \models \phi$ is equivalent to universally verifying $R_h, (r,0), \{O_1(s_e(r,0)),...,O_n(s_e(r,0))\} \models \phi$ for all $r \in R_h$.

The algorithm proceeds from time 0 to time $O(k\cdot|\phi|)$. When dealing with a relation $R_h, (r,k), \{v_1,...,v_n\} \models \phi$, it needs $s(M,Act,\phi,k) = O(n \cdot |\phi| \cdot k \cdot (\log|M| + \log|Act|))$ bits to remember the observation history $\{v_1,...,v_n\}$, where n is the number of players, $|\phi| \cdot k$ is the maximal length of observation history, and $\log|M| + \log|Act|$ is the number of bits to store an observation. The whole recursive procedure needs $a(\phi,k) = O(|\phi| \cdot k)$ alternations.

By Theorem 4.2 of [9], the algorithm can be simulated by a deterministic machine using $a(\phi,k)s(M,Act,\phi,k) + s(M,Act,\phi,k)^2 = O(|\phi|\cdot k \cdot n \cdot |\phi| \cdot k \cdot (\log|M| + \log|Act|) + (n \cdot |\phi| \cdot k \cdot (\log|M| + \log|Act|))^2)$ space. Therefore, we have the upper bound of PSPACE.

The lower bound can be obtained by a reduction from the satisfiability problem of quantified Boolean formulas (QBF). The detail is omitted here for space limit. □

The result is interesting by its own. Although two player reachability game is EXPTIME-complete [26], the reachability game of polynomial steps is strictly simpler in PSPACE-complete.

8. A SYMBOLIC OBDD-BASED BOUNDED MODEL CHECKING ALGORITHM

While in general, model checking ATL of incomplete information and synchronous perfect recall is believed to be undecidable, we have shown that one of its fragments, the bounded ATL, is PSPACE-complete. In this section, we put forward a symbolic model checking algorithms based on OBDD's for the bounded ATL.

8.1 OBDD

An OBDD is a compact representation of a boolean function. It has been widely used in model checking for its space efficiency in storing functions, e.g., transition relation. The benefit of OBDD is that most useful operations, e.g., boolean operations, on OBDDs are efficient.

8.2 Algorithm

Given a formula ϕ, we define $d(\phi)$ to be its maximal temporal depth such that

- $d(p) = 0$, $d(\neg\phi) = d(\phi)$, $d(\phi_1 \wedge \phi_2) = max\{d(\phi_1), d(\phi_2)\}$
- $d(\langle\langle A\rangle\rangle X\phi) = d(\phi) + 1$
- $d(\langle\langle A\rangle\rangle\phi_1 \, U^{\leq k}\phi_2) = max\{d(\phi_1), d(\phi_2)\} + k$
- $d(\langle\langle A\rangle\rangle\phi_1 \, R^{\leq k}\phi_2) = max\{d(\phi_1), d(\phi_2)\} + k$

Let the size of formula be the number of operators and the size of model be the number of states. The maximal temporal depth is $O(|\phi| \cdot k)$.

Let $as_e[0..d](r) = s_e(r,0)a(r,1)...a(r,d)s_e(r,d)$ be the run prefix of $r \in \mathcal{R}$ up to time d. We write $\mathcal{R}_d(M)$ for the set of run prefixes up to time d, i.e.,

$$\mathcal{R}_d(M) = \{as_e[0..d](r) \mid r \in \mathcal{R}\}.$$

DEFINITION 2. *Given a formula ϕ, we define a function $f(\varphi, x) : \mathcal{R}_{d(\phi)}(M) \to \{0,1\}$, to be the encoding of the subformula φ of ϕ holds at time x of runs with prefix $t \in \mathcal{R}_{d(\phi)}(M)$. These values can be computed recursively by the rules as follows.*

- $f(p,x)(t) = p \in \pi(t(x))$
- $f(\neg\varphi, x)(t) = \neg f(\varphi, x)(t)$
- $f(\varphi_1 \wedge \varphi_2, x)(t) = f(\varphi_1, x)(t) \wedge f(\varphi_2, x)(t)$
- $f(\langle\langle A\rangle\rangle\varphi_1 \, U^{\leq k}\varphi_2, x)(t) =$

$$\begin{cases} f(\varphi_2 \vee (\varphi_1 \wedge \langle\langle A\rangle\rangle X(\langle\langle A\rangle\rangle\varphi_1 \, U^{\leq k-1}\varphi_2)), x)(t) & k > 0 \\ f(\varphi_2, 0)(t) & k = 0 \end{cases}$$

- $f(\langle\langle A\rangle\rangle\varphi_1 R^{\leq k}\varphi_2, x)(t) =$

$$\begin{cases} f(\varphi_2 \wedge (\varphi_1 \vee \langle\langle A\rangle\rangle X(\langle\langle A\rangle\rangle\varphi_1 R^{\leq k-1}\varphi_2)), x)(t) & k > 0 \\ f(\varphi_2 \wedge \varphi_1, 0)(t) & k = 0 \end{cases}$$

- $f(\langle\langle A\rangle\rangle X\varphi, x)(t) = \exists a_A :$
$((\exists t' \in \mathcal{R}_d(M) : viewequ_A(t, t', x) \wedge a_A(t'(x+1)) = a_A) \wedge$
$(\forall t' \in \mathcal{R}_d(M) : viewequ_A(t, t', x) \wedge a_A(t'(x+1)) = a_A \Rightarrow$
$f(\varphi, x+1)(t')))$

where

- $viewequ_A(t, t', x) = \bigwedge_{j=0}^{x} O_A(t(j)) = O_A(t'(j)) \wedge \bigwedge_{j=1}^{x} a_A(t(j)) = a_A(t'(j))$

The following theorem characterizes model checking using the function $f(\varphi, x)$.

THEOREM 2. *Let M be any finite PO-CGS and ϕ a bounded ATL formula, we have $M \models \phi$ iff $\forall t \in \mathcal{R}_d(M) : (f(\phi, 0)(t) = 1)$, where $d = d(\phi)$.*

8.3 Correctness

We show the correctness of theorem 2. First of all, we claim that if two strategies result in the same view in a finite number of steps, then they will enforce the same goals since then. The main intuition is that, every future step will involve a strategic update and both of the strategies can be updated into a same strategy.

Let $E_A(r, m, a_A) = \{r' \in \mathcal{R} \mid view_A(r', m) = view_A(r, m) \wedge a_A(r', m) = a_A\}$ be a set of runs that 1) have the same view as that of run r up to time m and 2) take the action $a_A \in Act_A$. Let $a_A^c(r, m)$ be the action that is taken by players in A on cell c if their view is consistent with that of point (r, m).

LEMMA 1. *Let (r, m) be a point in $\mathcal{I}$ and a_A be an action of players in A such that $E_A(r, m, a_A) \neq \emptyset$. Let $[c_1]_C^{\simeq_A}$ and $[c_2]_C^{\simeq_A}$ be two equivalence classes of an AIS $\mathcal{I}$ such that $[c_1]_C^{\simeq_A} \cap E_A(r, m, a_A) \neq \emptyset$ and $[c_2]_C^{\simeq_A} \cap E_A(r, m, a_A) \neq \emptyset$. Then we have the following equivalence:*

- $\forall r_1' \in [c_1]_C^{\simeq_A} \cap E_A(r, m, a_A) : \mathcal{I}, (r_1', m+1) \models \phi$
- $\forall r_2' \in [c_2]_C^{\simeq_A} \cap E_A(r, m, a_A) : \mathcal{I}, (r_2', m+1) \models \phi$.

PROOF. (Sketch) Firstly, we note that for any cell c, if $[c]_C^{\simeq_A} \cap E_A(r, m, a_A) \neq \emptyset$ then $a_A^c(r, m) = a_A$. Therefore, we have $a_A^{c_1}(r, m) = a_A = a_A^{c_2}(r, m)$.

Now let $F_A(r, m, c) = \{s_e(r', m) \mid r' \in [c]_C^{\simeq_A} \cap E_A(r, m, a_A^c(r, m))\}$. We show that $F_A(r, m, c_1) = F_A(r, m, c_2)$. It is proved by induction on m. For the base case, $F_A(r, 0, c_1) = F_A(r, 0, c_2) = I$. Assume that $F_A(r, k, c_1) = F_A(r, k, c_2)$ for $0 \leq k \leq m$. Note that $F_A(r, k+1, c_1) = \{s' \mid s \in F_A(r, k, c_1), \exists a_{\bar{A}} \in Act_{Agt\setminus A} : s' \in T(s, a_A^{c_1}(r, k) a_{\bar{A}})\}$, which by $a_A^{c_1}(r, k) = a_A^{c_2}(r, k)$ and $F_A(r, k, c_1) = F_A(r, k, c_2)$, is equivalent to $\{s' \mid s \in F_A(r, k, c_2), \exists a_{\bar{A}} \in Act_{Agt\setminus A} : s' \in T(s, a_A^{c_2}(r, k) a_{\bar{A}})\}$. Therefore, we have $F_A(r, k+1, c_1) = F_A(r, k+1, c_2)$.

Finally, by the syntax of ATL, except for boolean operators, all operators in ϕ can only be coalition operators. Each coalition operator involves a change on the strategy and thus the future behaviours are irrelevant to the previous strategy. Combining with $F_A(r, m, c_1) = F_A(r, m, c_2)$, we have the conclusion. □

LEMMA 2. *Let M be any finite PO-CGS, and $\mathcal{I} = (\mathcal{R}, C, \{\simeq_i\}_{i \in Agt}, \pi)$ be an interpreted system constructed from M and synchronous perfect recall. For any formula φ of bounded ATL and any point $(r, m) \in \mathcal{R} \times \mathbb{N}$, we have that $\mathcal{I}, (r, m) \models \varphi$ iff $f(\varphi, m)(as_e[0..d](r)) = 1$.*

No. of Iterations	5	6	7	8	9
Specification (1)	52.1	197.4	402.5	1535.9	9140.8

Table 2: Running Times of Iterated Prisoner'd Dilemma

PROOF. (Sketch) We prove it by induction on the structure of the formula φ. Here we only deal with two cases, others can be obtained similarly.

1) $\mathcal{I}, (r, m) \models p$ is equivalent to $p \in \pi(r, m) = \pi(r_e(m))$ (by Definition 1). By $m \leq d$, we have $r_e(m) = as_e[0..d](r)(m)$. Then by the algorithm, we have $p \in \pi(as_e[0..d](r)(m)) = f(p, m)(as_e[0..d](r))$.

2) $\mathcal{I}, (r, m) \models \langle\langle A\rangle\rangle X\varphi'$ is equivalent to $\exists [c]_C^{\simeq_A} \in [\simeq_A]_C : (\exists r' \in [c]_C^{\simeq_A} : ((r', m) \in \mathcal{K}_A(r, m))) \wedge (\forall r' \in [c]_C^{\simeq_A} : (r', m) \in \mathcal{K}_A(r, m) \Rightarrow \mathcal{I}, (r', m+1) \models \varphi')$ (by Definition 1). Let a_A be the action taken by the players A in cell $[c]_C^{\simeq_A}$ for their view that is consistent with point (r, m). Then the later is equivalent to $\exists [c]_C^{\simeq_A} \in [\simeq_A]_C : \exists a_A \in Act_A : (E_A(r, m, a_A) \cap [c]_C^{\simeq_A} \neq \emptyset) \wedge (\forall r' \in E_A(r, m, a_A) \cap [c]_C^{\simeq_A} : \mathcal{I}, (r', m+1) \models \varphi')$, which by Lemma 1, is equivalent to $\exists a_A \in Act_A : (\exists [c]_C^{\simeq_A} \in [\simeq_A]_C : E_A(r, m, a_A) \cap [c]_C^{\simeq_A} \neq \emptyset) \wedge (\forall r' \in E_A(r, m, a_A) : \mathcal{I}, (r', m+1) \models \varphi')$. Finally, because a_A is defined in the cell $[c]_C^{\simeq_A} \in [\simeq_A]_C$, we have that $\exists a_A \in Act_A : (E_A(r, m, a_A) \neq \emptyset) \wedge (\forall r' \in E_A(r, m, a_A) : \mathcal{I}, (r', m+1) \models \varphi')$. Then by recursive hypothesis and the algorithm, we have $f(\langle\langle A\rangle\rangle X\varphi', m)(t) = 1$. □

Proof of Theorem 2 Let $\mathcal{I} = (\mathcal{R}, C, \{\simeq_i\}_{i \in Agt}, \pi)$ be the interpreted system constructed from M and synchronous perfect recall. By the definition, $M \models \phi$ is equivalent to $\forall r \in \mathcal{R} : (\mathcal{I}, (r, 0) \models \phi)$, which by Lemma 2, is equivalent to $\forall r \in \mathcal{R} : (f(\phi, 0)(as_e[0..d](r)) = 1)$. The latter is equivalent to $\forall t \in \mathcal{R}_d(M) : (f(\phi, 0)(t) = 1)$ by the definition of $\mathcal{R}_d(M)$. □

9. APPLICATIONS

The algorithm has been implemented in the epistemic model checker MCK [11]. As our experiments, it is used to verify several applications. All experimental data are collected on an Apple iMac with core i3 CPU and 4G memory. Running times are measured in seconds.

9.1 Iterated Prisoner's Dilemma

First of all, the model checking results justify our analysis for the three specifications on IPD(3). To show the capability of the algorithm in dealing with IPD, we scale up the number of iterations k. As the three specifications need similar running times for a given k, we only report the results for the specification (1) in Table 2

9.2 Kriegspiel Tic-Tac-Toe

The game of Kriegspiel Tic-Tac-Toe [27] is played on a $n \times n$ board, as in Figure 2. Two players × and ◯ take turns, by marking X and O on the squares of the board, respectively. The first player to occupy a horizontal, vertical, or diagonal row wins the game. Players can't see where their opponent plays. If they try to mark a square that has been marked by their opponent, they are informed of this fact but not allowed to play again.

The game state is a pair (x, o) of boolean variables, denoting whether in the current round, players × and ◯ have successfully marked the squares they are requesting for. Each of them can only observe part of a system state, i.e., $O_\times(x, o) = x$ and $O_\bigcirc(x, o) = o$. Here we make a remark that, players do not need to explicitly keep tracking the board by $n \times n$ variables. The status of the board can be obtained from a sequence of games states and local actions observed from the past rounds.

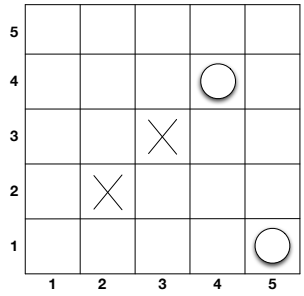

Figure 2: An 5×5 board for Kriegspiel Tic-Tac-Toe

Board ($n \times n$)	3×3	4×4	5×5	6×6
k = 6	11.7	19.6	211.3	2470.3
k = 9	19.8	25.5	432.4	3315.7
k = 12	86.9	135.1	477.1	4032.7

Table 3: Running Times of Kriegspiel Tic-Tac-Toe ($n \times n$)

Let $m_{x,y}$ be the action of marking the square (x, y). An action of a player is to mark a square in the board, that is, $a_\times, a_\bigcirc \in \{m_{x,y} \mid (x, y) \in board\}$. A global action is of the form $(m_{x,y}, nil)$ or $(nil, m_{x,y})$, depending on which player is taking a turn, where *nil* denotes an empty action.

In this game, we want to check if one of the players has a winning strategy in a bounded number of rounds, e.g.,

$$\langle\!\langle \{\times\} \rangle\!\rangle True\ U^{\leq k}\ xwin \tag{4}$$

where the atomic proposition *xwin* expresses that the player $\times$ wins the game. The experiments are conducted by scaling up both the size of board and the number of rounds. Table 3 gives the running time results.

9.3 Patrolling Game

Growing interests has been accumulated on the study of *patrolling problem* from the game theoretical view [5]. This line of research has resulted in successful applications in, e.g., the development of security scheduler for the Los Angeles International Airport. Most of the works focus on developing a designated algorithm to synthesise a strategy. We here propose a way to investigate a patrolling game from the view of model checking, by following the notations of [17, 19].

There are two players, a guard G and an attacker A. The guard G patrols the area by following a specific plan. The attacker A can succeed if holding an attack at a position for a continuous z rounds without being captured by G. Moreover, an attack, once started, can't be terminated and thus will always result in a successful attack or a capture.

Let $G = (V, E)$ be a discrete graph consisting of a set V of positions and a set E of edges connecting positions. A game state is a tuple $(pos_{\mathtt{G}}, pos_{\mathtt{A}}, counter, capture)$, where $pos_{\mathtt{G}}$ is current position of G, $pos_{\mathtt{A}}$ is current position of A, *counter* shows the elapsed time of an attack, and *capture* denotes whether A has been captured by G. The observation functions are defined as

$$O_{\mathtt{G}}((pos_{\mathtt{G}}, pos_{\mathtt{A}}, counter, capture)) = (pos_{\mathtt{G}}, capture)$$

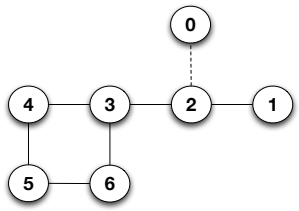

Figure 3: A Graph from Patrolling Games

Graph (G)	k=5	k=6	k=7	k=8	k=9
Specification (5)	42.3	209.2	708.8	2275.5	10170.3

Table 4: Running Times of Patroling Game ($G = (n, m)$)

and

$$O_{\mathtt{A}}((pos_{\mathtt{G}}, pos_{\mathtt{A}}, counter, capture)) = \begin{cases} (pos_{\mathtt{G}}, pos_{\mathtt{A}}, counter, capture) & \text{if } (pos_{\mathtt{G}}, pos_{\mathtt{A}}) \in E \\ (nil, pos_{\mathtt{A}}, counter, capture) & \text{otherwise} \end{cases}$$

which means that the attacker A can observe its neighbouring nodes.

At each time, the available actions for a player is to stay at its current position or move to an adjacent position if they are connected.

In such a game, an interesting specification is

$$\langle\!\langle \{\mathtt{A}\} \rangle\!\rangle X^k (counter = z \wedge \neg capture) \tag{5}$$

which expresses that the attacker A has a strategy to hold a successful attack, or its negation which says that the guard G can guarantee the security of the area.

In our experiments, we assume an area as described in Figure 3. The area consists of 6 positions {1..6} and an outside position 0. The attacker starts in 0 and the guard can start at any position except for 0. The guard plays by following a specific strategy: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 3 \to 2 \to 1$.

Assume that the attack time is $z \leq 4$. A strategy for the attacker is that, each time it sees the guard at position 2, it moves to position 2 at the next time. If the guard appears at position 1, then it moves back to position 0. Otherwise, it starts attacking position 2. Because the guard can only return to position 2 after 5 rounds, the attacker can make a successful attack.

If $5 \leq z < 7$, the attacker can still make a successful attack by following the guard to the circle area, i.e., positions in $\{4, 5, 6\}$, and then start attacking. On the other hand, the attack will absolutely fail if $z \geq 7$. The experimental results are shown in Table 4 by scaling up the number k of rounds. The setting of z does not affect the running times.

10. RELATED WORKS

ATL logic has attracted many attentions in recent years. Here we only review some closely-related works. For complete information systems, the complexity of verifying ATL logic has been extensively studied, for both memoryful strategies [2, 8] and memoryless strategies [2, 22]. See e.g., [7] for an overview. Mocha [1] is a well-known model checker.

For incomplete information systems, the complexity of observational view has been studied [28] and the complexity of perfect recall view is believed to be undecidable in general [2]. The model checker MCMAS [23] can deal with formulas of observational view, with the semantics and the algorithm presented in [24].

11. CONCLUSION

We have presented a symbolic OBDD-based algorithm to model checking bounded ATL in systems of incomplete information and synchronous perfect recall. The computational complexity of model checking bounded ATL is shown to be PSPACE-complete. The symbolic algorithm is implemented in an epistemic model checker MCK and the experimental results show its usefulness in deal with interesting applications.

There are two directions that we will explore in the future. The first is to improve the performance of the algorithm, by considering the techniques developed for temporal epistemic logics [29, 16]. The second is to extend the algorithm to work with stochastic multiagent systems. We have proposed PATL logic [18, 14] to express the properties like "a set A of agents have a collective strategy to enforce the goal ϕ in a probability more than 90%".

Acknowledgement

The author thanks Ron van der Meyden for his detailed and useful comments on a previous version of the paper.

12. REFERENCES

[1] Rajeev Alur, Luca de Alfaro, Radu Grosu, Thomas A. Henzinger, M. Kang, Christoph M. Kirsch, Rupak Majumdar, Freddy Y. C. Mang, and Bow-Yaw Wang. JMOCHA: A Model Checking Tool that Exploits Design Structure. In *ICSE 2001*, pages 835–836, 2001.

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[3] Robert Axelrod. Effecgive choice in the prisoner's dilemma. *Journal of Conflict Resolution*, 24(1):3, 1980.

[4] Robert Axelrod. *Evolution of cooperation*. Basic Books, 1984.

[5] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184-185:78–123, 2012.

[6] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58:118–149, 2003.

[7] Nils Bulling, Jurgen Dix, and Wojciech Jamroga. Model Checking Logics of Strategic Ability: Complexity. In *Specification and Verification of Multi-Agent Systems*. Springer, 2010.

[8] Nils Bulling and Wojciech Jamroga. Verifying agents with memory is harder than it seemed. *AI Communications*, 23(4):389–403, 2010.

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

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

[11] 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.

[12] Fausto Giunchiglia and Paolo Traverso. Planning as Model Checking. In *Recent Advances in AI Planning, European Conf. on Planning, ECP'99*, pages 1–20, 1999.

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

[14] Xiaowei Huang and Cheng Luo. A logic of Probabilistic Knowledge and Strategy. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2012)*, 2012.

[15] Xiaowei Huang, Cheng Luo, and Ron van der Meyden. Improved bounded model checking for a fair branching-time temporal epistemic logic. In *Sixth Workshop on Model Checking and Artificial Intelligence (MoChArt 2010)*, LNCS 6572, pages 95–111, 2010.

[16] Xiaowei Huang, Cheng Luo, and Ron van der Meyden. Symbolic Model Checking of Probabilistic Knowledge. In *13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK XII)*, pages 177–186, 2011.

[17] Xiaowei Huang, Patrick Maupin, and Ron van der Meyden. Model checking knowledge in pursuit-evasion games. In *the 22nd International Joint Conference on Artificial Intelligence (IJCAI2011)*, pages 240–245, 2011.

[18] Xiaowei Huang, Kaile Su, and Chenyi Zhang. Probabilistic Alternating-Time Temporal Logic of Incomplete Information and Synchronous Perfect Recall. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, pages 765–771, 2012.

[19] Xiaowei Huang and Ron van der Meyden. Synthesizing Strategies for Epistemic Goals by Epistemic Model Checking: an application to Pursuit-Evasion Games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, pages 772–778, 2012.

[20] Wojciech Jamroga. Strategic Planning through Model Checking of ATL Formulae. In *Artificial Intelligence and Soft Computing - ICAISC 2004*, pages 879–884, 2004.

[21] Wojciech Jamroga and Wiebe van der Hoek. Agents that Know How to Play . *Fundamenta Informaticae*, 62:1–35, 2004.

[22] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. On the expressiveness and complexity of atl. *Logical Methods in Computer Science*, 4(2), 2008.

[23] 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.

[24] Alessio Lomuscio and Franco Raimondi. Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In *5th international joint conference on Autonomous agents and multiagent systems (AAMAS 2006)*, pages 161–168, 2006.

[25] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. In *AAMAS*, pages 209–216. ACM, 2003.

[26] J. H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Science*, 29(2):274–301, 1984.

[27] Stephan Schiffel and Michael Thielscher. Reasoning About General Games Described in GDL-II. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 846–851, 2011.

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

[29] R. van der Meyden and Kaile Su. Symbolic Model Checking the Knowledge of the Dining Cryptographers. In *Proc. 17th IEEE Workshop on Computer Security Foundations*, pages 280–291, 2004.