Model Checking Probabilistic Epistemic Logic for Probabilistic Multiagent Systems

Chen Fu[1,2], **Andrea Turrini**[1], **Xiaowei Huang**[3], **Lei Song**[4], **Yuan Feng**[5], **Lijun Zhang**[1,2]
[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Department of Computer Science, University of Liverpool
[4] JD.com
[5] Centre for Quantum Software and Information, University of Technology Sydney
{fchen,turrini,zhanglj}@ios.ac.cn, Xiaowei.Huang@liverpool.ac.uk,
songlei3@jd.com, Yuan.Feng@uts.edu.au

Abstract

In this work we study the model checking problem for probabilistic multiagent systems with respect to the probabilistic epistemic logic PETL, which can specify both temporal and epistemic properties. We show that under the realistic assumption of uniform schedulers, i.e., the choice of every agent depends only on its observation history, PETL model checking is undecidable. By restricting the class of schedulers to be memoryless schedulers, we show that the problem becomes decidable. More importantly, we design a novel algorithm which reduces the model checking problem into a mixed integer non-linear programming problem, which can then be solved by using an SMT solver. The algorithm has been implemented in an existing model checker and experiments are conducted on examples from the IPPC competitions.

1 Introduction

Multiagent systems have found many applications in practice [Seuken and Zilberstein, 2008], ranging from coordinated helicopter flights [Nair *et al.*, 2005], to distributed sensors [Pynadath and Tambe, 2002]. In this paper, we focus on probabilistic multiagent systems which can model uncertainty, such as component failure, of the agents and/or their environment.

In multiagent systems it is typical that agents share incomplete information with each other, otherwise multiagent systems will degenerate to systems with a single agent [Kazmierczak *et al.*, 2014]. The incompleteness of information is normally characterized by defining for each agent i an equivalence relation $\sim_i$ over all global states of the systems; two global states are considered indistinguishable for a given agent i if they are related by $\sim_i$. Two states that are indistinguishable for an agent may be distinguishable for another agent. It is a natural, and realistic, setting that every agent makes its own decisions based only on the limited information it has; namely, the information provided by its own indistinguishable relation, and nothing else. Decisions of agents are usually formalized by the notion of *schedulers* (also known as policies or strategies), which are functions taking history executions as input and deciding the next move for each agent. Schedulers only making use of limited information of each agent are called *uniform* (or *decentralized*).

To specify the property of probabilistic multiagent systems, in particular the temporal dynamics of agents' knowledge, we use Probabilistic Epistemic Temporal Logic (PETL) (cf. [Delgado and Benevides, 2009]). PETL can be seen as a combination of epistemic logic [Fagin *et al.*, 2004] and probabilistic computation tree logic (PCTL) [Jonsson and Larsen, 1991]. For its semantics, the knowledge operator $\mathbf{K}_i$ is defined over agent i's indistinguishable relation and the probabilistic operator $[\psi]^{\bowtie d}$ is defined over uniform schedulers.

We study the model checking problem [Baier and Katoen, 2008] for PETL logic, with the following contributions. The first contribution is a set of complexity results, including the undecidability for the general case and the decidability (in P^{NP} and NP-hard) when schedulers are memoryless.

The second contribution is a symbolic model checking algorithm for memoryless schedulers. The algorithm reduces the model checking problem to a mixed integer non-linear programming problem, which can then be solved with a modern SMT solver such as Z3. The reduction procedure is novel, particularly for the part handling loops in the state space. While both strategic and epistemic reasoning on probabilistic multiagent systems have been studied in theory (mainly semantics), detailed works on the algorithmic design, implementation, and validation based on an existing model checker is not available. This paper presents *the first tool* to enable PETL model checking.

Finally, we validate our tool on models from the International Probabilistic Planning Competitions (IPPC). The experimental results show the scalability of our symbolic algorithm in handling interesting problems.

1.1 Related Work

In the non-probabilistic multiagent systems, schedulers with incomplete information have been studied extensively; see for instance [Jamroga, 2003; Jamroga and van der Hoek, 2004]. In [van der Hoek and Wooldridge, 2002], a model checking algorithm is presented for alternating temporal epistemic logic and it is shown that the problem can be reduced to model checking alternating temporal logic [Alur *et al.*, 2002].

The logic PETL was introduced in [Delgado and Benevides, 2009]. However, their semantics is defined under all schedulers, so the corresponding model checking problem has the same complexity with the standard PCTL one. [Huang and Luo, 2013] proposes a probabilistic epistemic logic pATEL*, which can be seen as a combination of probabilistic Alternating Time Logic (pATL) [Huang *et al.*, 2012] and epistemic logic. Different from [Schnoor, 2010], in [Huang and Luo, 2013] the epistemic accessibility relations are also probabilistic, namely, there is a probabilistic distribution over the set of indistinguishable states. Most of the existing works are theoretical, without detailed algorithmic design, implementation, and validation. One exception is [Huang *et al.*, 2011], which presents a symbolic model checking algorithm for fully probabilistic systems, i.e., systems without nondeterministic choices. Other exceptions are [Wan *et al.*, 2013; Sultan *et al.*, 2014], which also consider fully probabilistic systems. Different from [Huang *et al.*, 2011], [Wan *et al.*, 2013; Sultan *et al.*, 2014] convert epistemic probabilistic logic model checking problem to the standard PCTL one, and then make use of the PRISM model checker. However, their approach can not be extended to deal with our problem, because our problem is under uniform schedulers in probabilistic systems with nondeterministic choices.

Existing model checkers for multiagent systems include MCMAS [Lomuscio *et al.*, 2017], which does not work with probabilistic systems, and MCK [Huang and van der Meyden, 2014; Huang *et al.*, 2011], which cannot handle PETL model checking problem. Probabilistic model checkers such as PRISM [Kwiatkowska *et al.*, 2011] do not work with epistemic logics and incomplete information systems.

2 Preliminaries

Given a finite set X, a discrete probability *distribution* on X is a function $\mu\colon X \to [0,1]$ such that $\mu(X) = \sum_{x\in X} \mu(x) = 1$. Let $\mathrm{Disc}(X)$ denote the set of all discrete probability distributions over X. We denote by $\delta_x \in \mathrm{Disc}(X)$ the *Dirac* distribution such that $\delta_x(y) = 1$ if $y = x$, 0 otherwise.

2.1 Probabilistic Concurrent Game Structure

A PO-PCGS (partially observed probabilistic concurrent game structure) with n agents is a tuple

$$\mathcal{M} = (S, \bar{s}, Agt, \{Act_i\}_{i\in Agt}, \{\sim_i\}_{i\in Agt}, \mathcal{T}, AP, L)$$

where

- S is a finite set of states with initial state $\bar{s} \in S$;
- Agt is the set of n agents;
- Act_i is the set of actions for agent i. We denote by $Act = \prod_{i=1}^{n} Act_i$ the set of full actions;
- $\mathcal{T}\colon S \times Act \to \mathrm{Disc}(S)$ is a partial transition function;
- For each $i \in Agt$, $\sim_i$ is an accessibility equivalence relation on S. For $s \in S$, denote by $[s]_i$ the equivalence class of $\sim_i$ which contains s;
- AP is a finite set of atomic propositions; and
- $L\colon S \to 2^{AP}$ is a labeling function.

As notation, we let $r, s, t, \ldots$ and their variants with indices range over S, $a, b, \ldots$ range over Act_i, and $\alpha, \beta, \ldots$ range over Act. Given a state s and a full action α, we say that α is enabled by s if $\mathcal{T}(s, \alpha) = \mu$, also written $s \xrightarrow{\alpha} \mu$. For an agent $i \in Agt$, we denote by $EA_i(s) \subseteq Act_i$ the set of enabled actions by s for agent i. Similarly, we denote by $EA(s) \subseteq Act$ the set of enabled full actions at state s. We wirte $P(s, \alpha, t) = \mu(t)$ where $s \xrightarrow{\alpha} \mu$.

The equivalence relation $\sim_i$ for the agent i relates states that are considered to be not distinguishable by the agent i; a first consequence is that two related states must enable the same actions, i.e., whenever $s \sim_i t$, then $EA_i(s) = EA_i(t)$.

A path π is a finite or infinite sequence of alternating states and actions $s_0\alpha_0 s_1\alpha_1 \ldots$, starting (and ending, if finite) with a state, such that for each $k > 0$, $s_k \xrightarrow{\alpha_k} \mu_k$ and $\mu_k(s_{k+1}) > 0$. For $\pi = s_0\alpha_0 s_1\alpha_1 \ldots$ and $k \geq 0$, we denote by $\pi[k]$ the state s_k. If π is finite, then we denote by $|\pi|$ the size (or, length) of π, i.e., the number of states in π, and by $\widehat{\pi}\downarrow$ its last state. We denote the sets of finite and infinite paths by $Paths^*$ and $Paths^\omega$, respectively, and we let $\widehat{\pi}$ and π and their variants with indices range over $Paths^*$ and $Paths^\omega$, respectively.

In the following we assume that all states of an PO-PCGS $\mathcal{M}$ are reachable, i.e., for each state $s \in S$, there exists a finite path $\widehat{\pi} = s_0\alpha_0 s_1\alpha_1 \ldots s_n$ such that $s_0 = \bar{s}$ and $s_n = s$.

Schedulers of PO-PCGS

Schedulers (also known as policies, strategies, or adversaries) are functions that resolve nondeterminism, i.e., they are used to choose the next action to be performed (based on the past history) whenever we reach a state enabling multiple actions. Once nondeterministic choices are solved, we can define an appropriate probability measure for the events in a PO-PCGS, like reaching specific states.

A *scheduler* for an agent i is a function $\sigma_i\colon Paths^* \to Act_i$ mapping each finite path $\widehat{\pi} \in Paths^*$ to an action in $EA_i(\widehat{\pi}\downarrow)$. A *uniform scheduler* σ_i for an agent i is a scheduler such that for each pair of finite paths $\widehat{\pi} = s_0\alpha_0 s_1\alpha_1 \ldots s_n$ and $\widehat{\pi}' = s'_0\alpha'_0 s'_1\alpha'_1 \ldots s'_n$, it holds that if for each $0 \leq k \leq n$, $\alpha_k = \alpha'_k$ and $s_k \sim_i s'_k$, then $\sigma_i(\widehat{\pi}) = \sigma_i(\widehat{\pi}')$. A *uniform memoryless scheduler* for an agent i is a scheduler such that for all states s, t, we have that $s \sim_i t$ implies $\sigma_i(s) = \sigma_i(t)$.

A scheduler σ for a PO-PCGS $\mathcal{M}$ with n agents is a tuple $(\sigma_1, \ldots, \sigma_n)$ such that each σ_i is a scheduler for agent i. We say that σ is uniform (memoryless) if each σ_i is uniform (memoryless) and we denote by $\mathcal{S}_U$ and $\mathcal{S}_{UM}$ the sets of uniform and uniform memoryless schedulers, respectively.

A scheduler σ and a state s induce a unique probability measure $Pr_{\sigma,s}$ on the σ-field generated by cylinder sets of finite paths. Intuitively, the probability of (the cylinder set of) a finite path $\pi = s_0\alpha_0 s_1\alpha_1 \ldots s_n$ is the product of the probability of going from s_k to s_{k+1} given that σ has chosen α_k, for each k. See [Baier and Katoen, 2008] for more details.

2.2 Probabilistic Epistemic Temporal Logic

In this section we present the syntax and semantics of Probabilistic Epistemic Temporal Logic (PETL) initially introduced in [Delgado and Benevides, 2009] as K-PCTL. PETL formulas are defined according to the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\psi]^{\bowtie d} \mid \mathbf{K}_i\varphi \mid \mathbf{E}_G\varphi \mid \mathbf{D}_G\varphi \mid \mathbf{C}_G\varphi$$
$$\psi ::= \mathbf{X}\varphi \mid \varphi \,\mathbf{U}\, \varphi$$

where $\bowtie \in \{<, >, \geq, \leq\}$, $d \in [0, 1]$, $i \in Agt$, $p \in AP$, and $G \subseteq Agt$. From the above grammar, we can see that PETL subsumes both PCTL and epistemic formulas.

Let $\sim_G^E = \bigcup_{i\in G} \sim_i$, $\sim_G^D = \bigcap_{i\in G} \sim_i$, and $\sim_G^C$ be the transitive closure of $\sim_G^E$. Let s be a state of a PO-PCGS, and φ a PETL formula. We say that s satisfies φ, denoted $s \models \varphi$, iff

- $s \models p$ if $p \in L(s)$, $s \models \neg\varphi$ if $s \not\models \varphi$, and $s \models \varphi_1 \wedge \varphi_2$ if $s \models \varphi_1$ and $s \models \varphi_2$, as usual;
- $s \models \mathbf{K}_i\varphi$ iff $t \models \varphi$ for all $t \in [s]_i$;
- $s \models \mathbf{E}_G\varphi$ iff $t \models \varphi$ for all $t \sim_G^E s$;
- $s \models \mathbf{D}_G\varphi$ iff $t \models \varphi$ for all $t \sim_G^D s$;
- $s \models \mathbf{C}_G\varphi$ iff $t \models \varphi$ for all $t \sim_G^C s$;
- $s \models [\psi]^{\bowtie p}$ iff for all $\sigma \in \mathcal{S}_U$,
$$Pr_{\sigma,s}(\{\, \pi \in Paths^\omega \mid \pi \models \psi \,\}) \bowtie q.$$

where

- $\pi \models \mathbf{X}\varphi$ iff $\pi[1] \models \varphi$;
- $\pi \models \varphi_1 \,\mathbf{U}\, \varphi_2$ iff there exists $k \geq 0$ such that $\pi[k] \models \varphi_2$ and $\pi[j] \models \varphi_1$ for all $0 \leq j < k$.

Different from [Delgado and Benevides, 2009], the semantics of PETL in our paper is defined over all uniform schedulers instead of general schedulers. As we have discussed in the introduction, this restriction is natural for multiagent systems.

We want to remark that, while the probabilistic operator $[\psi]^{\bowtie d}$ makes use of the schedulers to evaluate the probability of the paths satisfying the path formula ψ, the agents are not aware of the current scheduler, so they can not make use of such an information in evaluating their knowledge. This means that the agents do not have to consider the potential unreachability problem caused by the choices made by the current scheduler (cf. [Jamroga and van der Hoek, 2004]).

3 Model Checking PETL

In this section, we discuss the problem of model checking PETL on PO-PCGSs. We show that the problem is in general undecidable but it becomes decidable if we restrict the schedulers to be memoryless. We then analyze the complexity of the restricted problem.

3.1 Undecidability of General Cases

Theorem 1. *The PETL model checking problem, restricted to uniform schedulers, is undecidable, even for PO-PCGSs with a single agent.*

Proof. The proof is by reduction of the undecidable problem about the emptiness of a Probabilistic Finite Automaton (PFA) [Rabin, 1963; Paz, 1971] to model checking PETL.

The reductions works as follows: given a PFA $\mathcal{A}$, it is considered as a PO-PCGS $\mathcal{M}_\mathcal{A}$ with a single agent that is only able to distinguish final from non-final states; each state is labelled by f if and only if it is final. Then it is possible to show that, given $\mathcal{A}$ and $\rho \in (0, 1)$, there exists a finite word $w \in \Sigma^*$ such that $\mathcal{A}(w) > \rho$ (i.e., w is accepted by $\mathcal{A}$ with probability at least ρ) if and only if $\bar{s} \not\models [tt \,\mathbf{U}\, f]^{\leq \rho}$. □

3.2 Complexity under Memoryless Setting

Theorem 2. *If restricted to uniform memoryless schedulers, model checking PETL is in P^{NP} and NP-hard. The NP-hardness holds even for PO-PCGSs with a single agent.*

P^{NP} Membership

Propositional formulas and epistemic formulas can be solved in PTIME. We can decide whether $s \models [\psi]^{\bowtie d}$ by:

1. Guessing a scheduler $\sigma \in \mathcal{S}_{UM}$;
2. Deciding whether $Pr_{\sigma,s}(\{\, \pi \in Paths^\omega \mid \pi \models \psi \,\}) \bowtie d$, which is model checking PCTL on Markov chain and can be solved in PTIME [Baier and Katoen, 2008].

So $s \models [\psi]^{\bowtie d}$ can be solved in NP. However, the probabilistic operator can be nested, so a polynomial procedure with respect to the size of formula is needed to query a NP oracle, which results the PETL model checking problem in P^{NP}.

NP-hardness

We reduce the 3SAT problem to this problem. Let $X = \{x_0, \ldots, x_k\}$ be a set of Boolean variables. An assignment is a total function $\theta \colon X \to \{0, 1\}$. Let a clause C be the disjunction of three Boolean variables or their negations. A formula is a conjunction of clauses.

The following problem is well-known to be NP-complete: given any Boolean formula $f = C_0 \wedge \cdots \wedge C_l$, decide whether there exists an assignment under which f is true.

Given a Boolean formula f, we encode it as a PO-PCGS $\mathcal{M} = (S, \bar{s}, \{i\}, Act, \{\sim_i\}, \mathcal{T}, AP, L)$ as follows:

- $S = \{\bar{s}\} \cup \{\, s_{m,n}^b \mid b \in \{0, 1\} \wedge 0 \leq m \leq k + 1 \wedge 0 \leq n \leq l \,\}$;
- $Act = \{\alpha, \alpha^0, \alpha^1\}$ and there is only one agent i;
- $\sim_i$ is defined as:
  - $s_{m,n}^b \sim s_{m,n'}^{b'}$ for all $b, b' \in \{0, 1\}$, $0 \leq m \leq k$, and $0 \leq n, n' \leq l$;
  - $s_{k+1,n}^b \sim s_{k+1,n'}^b$ for all $b \in \{0, 1\}$, $0 \leq n, n' \leq l$;
- $\mathcal{T}$ is defined as follows:
  - $\mathcal{T}(\bar{s}, \alpha) = \mu$ where $\mu(s_{0,n}^0) = \frac{1}{l+1}$ for $0 \leq n \leq l$;

- for each $0 \leq m \leq k$, $0 \leq n \leq l$, and $b \in \{0,1\}$:

$$\mathcal{T}(s^0_{m,n}, \alpha^0) = \begin{cases} \delta_{s^1_{m+1,n}} & \neg x_m \in C_n \\ \delta_{s^0_{m+1,n}} & \neg x_m \notin C_n, \end{cases}$$

$$\mathcal{T}(s^0_{m,n}, \alpha^1) = \begin{cases} \delta_{s^1_{m+1,n}} & x_m \in C_n \\ \delta_{s^0_{m+1,n}} & x_m \notin C_n, \end{cases}$$

$$\mathcal{T}(s^1_{m,n}, \alpha^b) = \delta_{s^1_{m+1,n}}$$

$$\mathcal{T}(s^b_{k+1,n}, \alpha) = \delta_{s^b_{k+1,n}}$$

- $AP = \{\top\}$; and
- L is defined as follows: for each $s \in S$,

$$L(s) = \begin{cases} \{\top\} & \text{if } s = s^1_{k+1,n} \text{ with } 0 \leq n \leq l, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

Then we have that f is NOT satisfiable iff $\bar{s} \models [tt\, \mathbf{U}\, \top]^{<1}$. The above reduction is obviously in polynomial time.

4 PETL Model Checking Algorithm

In the previous section we have seen that model checking PETL is in general undecidable but it becomes decidable if we restrict the schedulers to be uniform and memoryless. In this section, we present an algorithm, shown as Algorithm 1, to model check PETL against uniform memoryless schedulers in $\mathcal{S}_{UM}$.

In Algorithm 1, we use the following notation:

$$[s]_{\cup G} = \{\, s' \in S \mid \exists i \in G : s' \in [s]_i \,\},$$
$$[s]_{\cap G} = \{\, s' \in S \mid \forall i \in G : s' \in [s]_i \,\},$$

and $[s]_{(\cup G)^*}$ is the least fixed point of $f_{s,G}(X) = \{\, s' \in S \mid \exists t \in X \cup \{s\}, i \in G : s' \in [t]_i \,\}$. Note that the computation of the fixed point of $f_{s,G}$ requires at most $|S|$ iterations.

Algorithm 1 Procedure to compute the set of states satisfying a PETL formula φ

```
procedure $Sat(\varphi)$
    if $\varphi = p$ return $\{ s \in S \mid p \in L(s) \}$;
    if $\varphi = \neg\varphi'$ return $S \setminus Sat(\varphi')$;
    if $\varphi = \varphi_1 \wedge \varphi_2$ return $Sat(\varphi_1) \cap Sat(\varphi_2)$;
    if $\varphi = \mathbf{K}_i\varphi'$ return $\{ s \in S \mid [s]_i \subseteq Sat(\varphi') \}$;
    if $\varphi = \mathbf{E}_G\varphi'$ return $\{ s \in S \mid [s]_{\cup G} \subseteq Sat(\varphi') \}$;
    if $\varphi = \mathbf{D}_G\varphi'$ return $\{ s \in S \mid [s]_{\cap G} \subseteq Sat(\varphi') \}$;
    if $\varphi = \mathbf{C}_G\varphi'$ return $\{ s \in S \mid [s]_{(\cup G)^*} \subseteq Sat(\varphi') \}$;
    if $\varphi = [\mathbf{X}\varphi']^{\bowtie d}$ return $\{ s \in S \mid \forall \alpha \in EA(s) : \mathcal{T}(s,\alpha)(Sat(\varphi')) \bowtie d \}$;
    if $\varphi = [\varphi_1\, \mathbf{U}\, \varphi_2]^{\bowtie d}$ return $\{ s \in S \mid \forall \sigma \in \mathcal{S}_{UM} : Pr_{\sigma,s}(\{ \pi \in Paths^\omega \mid \pi \models \varphi_1\, \mathbf{U}\, \varphi_2 \}) \bowtie d \}$;
```

Since $\mathcal{S}_{UM}$ is finite, Algorithm 1 terminates: similarly to the model checking algorithm for PCTL on MDPs (see, e.g., [Baier and Katoen, 2008]), the algorithm follows a bottom-up approach based on the structure of the formula φ. All cases follow directly from the semantics; the interesting ones are about the probabilistic operator $[\,\cdot\,]^{\bowtie d}$ which has a temporal formula as argument. For the next operator $\mathbf{X}$, we can directly consider all actions enabled by the state since the type of schedulers does not affect the outcome. Instead, for the until operator $\mathbf{U}$, we have to consider the scheduler, since $\varphi_1\, \mathbf{U}\, \varphi_2$ may be satisfied by paths of different length. In the following section, we describe an algorithm to deal with all uniform memoryless schedulers at the same time.

4.1 Dealing with Uniform Memoryless Schedulers

In this section we show how to solve the problem of deciding whether $s \models [\varphi_1\, \mathbf{U}\, \varphi_2]^{\bowtie d}$ provided that we have computed $C = Sat(\varphi_1)$ and $B = Sat(\varphi_2)$, i.e., deciding whether $s \models [C\, \mathbf{U}\, B]^{\bowtie d}$, with respect of all uniform memoryless schedulers. In order to decide whether $s \models [C \mathbf{U} B]^{\bowtie d}$, it suffices to consider only $Pr^{\max}_s(\{\, \pi \in Paths^\omega \mid \pi \models C\, \mathbf{U}\, B \,\})$ and $Pr^{\min}_s(\{\, \pi \in Paths^\omega \mid \pi \models C\, \mathbf{U}\, B \,\})$ over all uniform memoryless schedulers.

Let's consider the minimal probability first. Let $Agt = \{1, \ldots, n\}$ and consider $x_s = Pr^{\min}_s(\{\, \pi \in Paths^\omega \mid \pi \models C\, \mathbf{U}\, B \,\})$. We write $s \models \exists\Diamond B$ to denote the fact that the set B is reachable from s in the underlying graph of $\mathcal{M}$. Then the vector $(x_s)_{s \in S} \in \mathbb{R}^{|S|}_{\geq 0}$ yields the unique solution of the following programming problem whose constraints are:

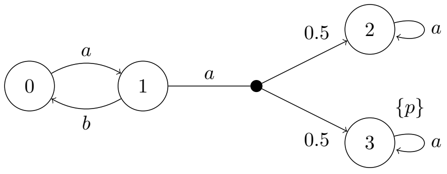
1. $x_s = 1$ if $s \in B$;
2. $x_s = 0$ if $s \not\models \exists\Diamond B$ or $s \notin C \setminus B$;
3. If $s \in C \setminus B$ and $s \models \exists\Diamond B$, then
$$x_s = \sum_{\substack{a_1 \in EA_1(s) \\ \cdots \\ a_n \in EA_n(s)}} p_{[s]_1,a_1} \cdot \ldots \cdot p_{[s]_n,a_n} \cdot \sum_{t \in S} P(s, a_1 \ldots a_n, t) \cdot x_t;$$
4. $p_{[t]_i,a_i} \in \{0,1\}$ for all $i \in Agt$, $t \in S$, and $a_i \in EA_i(t)$;
5. $\sum_{a_i \in EA_i(t)} p_{[t]_i,a_i} = 1$ for all $i \in Agt$ and $t \in S$;
6. $0 \leq x_s \leq 1$ for all $s \in S$,

where x_s is minimal.

Note that the above problem is a Mixed Integer Non-Linear Program (MINLP), while the programs of standard PCTL model checking are Linear Programs (LP).

Intuitively, having $p_{[s]_i,a_i} = 1$ represents the fact that agent i chooses action a_i at each state $t \in [s]_i$. Each uniform memoryless scheduler corresponds to one assignment for variables $p_{[s]_i,a_i}$. The minimal x_s is the minimal probability we want.

However, it is not straightforward to adapt the above program to compute the maximal probability $Pr^{\max}_s(\{\, \pi \in Paths^\omega \mid \pi \models C\, \mathbf{U}\, B \,\})$. We explain why with the help of the following PO-PCGS.



In this PO-PCGS, all states have empty label except for $L(3) = \{p\}$; there is only one agent who can distinguish every state. We want to compute $x_s = Pr_s^{\max}[tt\,\mathbf{U}\,p]$ for each state. Clearly, what we want to get is $x_0 = x_1 = 0.5$, $x_2 = 0$, and $x_3 = 1$. However, if we use the above program, when the agent chooses b at state 1, we get $x_0 = x_1$ and $x_1 = x_0$, so x_0, x_1 can be assigned to any value between 0 and 1. If we take the minimal value, we get 0; if we take the maximal value, we get 1: they are both different from the right answer.

The key part of this problem is that states 0 and 1 form a loop when the agent chooses b at state 1. In this case, x_0 and x_1 should be both 0, because they can not reach state 3. To force the value of a variable x_s to be 0 whenever the scheduler makes the state s unable to reach B, we extend the above problem by adding more constraints. We introduce a new variable f_s for each state s, so that $0 \leq f_s \leq |S|$, with the following meaning: $f_s = 0$ means that $s \not\models \exists\Diamond B$ or $s \notin C \setminus B$; $f_s \geq 1$ means that there is a path from s to B with length f_s. The corresponding constraints are as follows:

1. $f_s = 1$ if $s \in B$;
2. $f_s = 0$ if $s \not\models \exists\Diamond B$ or $s \notin C \setminus B$;
3. If $s \in C \setminus B$ and $s \models \exists\Diamond B$, then

$$\begin{aligned}
&f_s = 0 \implies x_s = 0\\
&f_s > 0 \implies \textstyle\bigvee_{t\in S} f_s = f_t + 1 \wedge P_{s,t} > 0\\
&f_s = 0 \iff \textstyle\bigwedge_{t\in S}(f_t = 0 \wedge P_{s,t} > 0) \vee P_{s,t} = 0\\
&f_s \neq 1 \wedge 0 \leq f_s \leq |S|
\end{aligned}$$

where

$$P_{s,t} = \sum_{\substack{a_1 \in EA_1(s)\\ \cdots \\ a_n \in EA_n(s)}} p_{[s]_1,a_1} \cdot \dots \cdot p_{[s]_n,a_n} \cdot P(s, a_1 \dots a_n, t).$$

Finally, we require that x_s is maximal.

Intuitively, $f_s = 0 \implies x_s = 0$ means that, if state s can not reach states in B under the current scheduler, then x_s must be 0. The constraint $f_s > 0 \implies \bigvee_{t\in S} f_s = f_t + 1 \wedge P_{s,t} > 0$ represents the fact that under the current scheduler, s can reach a successor t with positive probability $P_{s,t} > 0$ and $f_s = f_t + 1$, i.e., reaching B from s requires one step more than from t. The constraint $f_s = 0 \iff \bigwedge_{t\in S}(f_t = 0 \wedge P_{s,t} > 0) \vee P_{s,t} = 0$ ensures that s and each proper successor t of s have value 0, in case under the current scheduler they are not able to reach B. Finally, $f_s \neq 1 \wedge 0 \leq f_s \leq |S|$ ensures that f_s is finite and it is either 0 or an integer strictly larger than 1 (as byproduct of the equality $f_s = f_t + 1$ from the second constraint).

Consider again the previous example; we have the following constraints (after replacement and simplification):

$$\begin{aligned}
&f_3 = 1\\
&f_2 = 0\\
&f_0 \neq 1 \wedge 0 \leq f_0 \leq 4\\
&f_0 > 0 \implies f_0 = f_1 + 1\\
&f_0 = 0 \iff f_1 = 0\\
&f_1 \neq 1 \wedge 0 \leq f_1 \leq 4\\
&f_1 > 0 \implies \quad (f_1 = f_0 + 1 \wedge p_{1,b} = 1)\\
&\qquad\qquad\quad \vee (f_1 = f_2 + 1 \wedge p_{1,a} = 1)\\
&\qquad\qquad\quad \vee (f_1 = f_3 + 1 \wedge p_{1,a} = 1)\\
&f_1 = 0 \iff (f_0 = 0 \wedge p_{1,b} = 1) \wedge (p_{1,a} = 0)
\end{aligned}$$

size (row × col)	$\lvert S\rvert$	$\lvert\mathcal{T}\rvert$
2×3	52	506
2×4	97	996
2×5	156	1650
3×3	112	1204
3×4	196	2194
3×5	302	3476
4×5	720	9018
6×5	2080	27794

Table 1: Sizes of PCGS

If $p_{1,b} = 1$ (thus, $p_{1,a} = 0$), then we have $f_0 = f_1 = 0$ because there is no other assignment satisfying $0 \leq f_0 \leq 4 \wedge 0 \leq f_1 \leq 4 \wedge f_0 = f_1 + 1 \wedge f_1 = f_0 + 1$; if $p_{1,a} = 1$ (thus, $p_{1,b} = 0$), then we have $f_0 = 3$ and $f_1 = 2$.

5 Implementation and Experiments

We have implemented the proposed PETL model checking algorithm in ePMC [Hahn *et al.*, 2014] and we present the experiment results in this section. In our implementation, we use the SMT solver Z3 [de Moura and Bjørner, 2008] (version 4.6.0) to solve the MINLP problem. We have also tried to use CPLEX, a high-performance mathematical programming solver, which also supports convex MINLP problems; however it refuses to solve the MINLP problems we generate by stating that they are not convex. The models we use for our experiments are taken from the IPPC competitions held in 2011 and 2014, and the original IPPC domain decription of these models can be found at https://github.com/ssanner/rddlsim. All experiments were conducted on a 3.40GHz Intel Core i7-2600 CPU with 8GB of memory. The source code of our implementation is available at https://github.com/fuchen1991/epmc-petl.

5.1 Robot Navigation Model

In this model, there are several robots moving in a grid, trying to reach a goal area. Every cell makes the robots disappear with a different probability. Once the robot arrives at the goal, it will not disappear any more. Each robots has several actions available at each cell: moving left, right, up, and down. The robots are totally independent, namely, one robot does not know other robots' locations and actions. The properties we are concerned about are the probabilities of the robots getting to the goal or disappearing.

In this experiment, we have 2 robots; we varied the size of the grid by changing the number of rows and columns; the size of the resulting PO-PCGSs is summarized in Table 1.

Table 2 shows the time needed to compute the maximal probability of a robot eventually reaching its goal area. We can see that Z3 takes almost the whole execution time, so one can get better performance by using a faster solver.

Table 3 shows instead the time needed to get the minimum probability of a robot eventually reaching its goal area, which is 0. As we can see, computing minimal probability scales much better, since Z3 has fewer constraints to satisfy.

It is interesting to observe how the number of variables and constraints generated for computing $Sat([\varphi]^{\bowtie d})$ is loosely re-

size	variables	constraints	Z3 (s)	total (s)
2×3	152	523	8	9
2×4	262	941	127	128
2×5	400	1479	658	659
3×3	300	1088	446	447
3×4	496	1863	5410	5411

Table 2: Solving time for $Pr^{\max}[\mathbf{F}(\mathit{at\_goal1} \vee \mathit{at\_goal2})]$

size	variables	constraints	Z3 (s)	total (s)
2×3	100	268	0	0
2×4	165	449	2	2
2×5	244	672	2	3
3×3	188	512	0	1
3×4	300	828	3	4
3×5	434	1210	78	80
4×5	928	2636	76	78
6×5	2440	7060	561	565

Table 3: Solving time for $Pr^{\min}[\mathbf{F}(\mathit{at\_goal1} \vee \mathit{at\_goal2})]$

lated to the number of transitions $|\mathcal{T}|$ of the PO-PCGS (cf. Tables 1 and 2). This is due to the fact that a single constraint may correspond to several concrete transitions; for instance, the choice by the scheduler of each transition $s \xrightarrow{\alpha} \mu$ from a state s in $[t]_i$ with full action α involving the (agent) action a_i is represented by a single constraint $p_{[t]_i, a_i} \in \{0, 1\}$.

As formulas involving epistemic operators, consider $\Phi_1 = Pr^{\max}[\mathbf{G}(\mathbf{K}_{\mathit{robot1}}(\mathit{row1} \neq \mathit{row2} \vee \mathit{col1} \neq \mathit{col2}) \wedge \neg \mathit{disappeared1} \wedge \neg \mathit{disappeared2})]$ and $\Phi_2 = Pr^{\max}[\mathbf{G}(\mathbf{D}_{\mathit{robot1},\mathit{robot2}}(\mathit{row1} \neq \mathit{row2} \vee \mathit{col1} \neq \mathit{col2}) \wedge \neg \mathit{disappeared1} \wedge \neg \mathit{disappeared2})]$, requiring to compute the maximal probability of *robot1* (together with *robot2*, resp.) always knowing that the two robots are not in the same cell and that both robots have not disappeared, respectively. Table 4 shows the results for these two formulas on the instance with 2 rows and 5 columns, where in every cell the probability of disappearing lies in the interval $(0, 1)$, and the two robots have different goals and initial cells, with respect to three different accessibility relations: R_1 represents the fact that each robot is oblivious of the other robot; R_2 encodes the fact that the robots know whether they are in the same cell; and in R_3 each robot knows whether the other robot is at Manhattan distance at most 2.

First we note that for R_1, Z3 is not needed to solve Φ_1 because the robot does not know whether it is in the same initial cell with the other robot. Then, by giving the robots more knowledge, like by means of $\mathbf{D}_G$ or by the accessibility relations, then each robot knows some detail about where the other robot is so they can avoid to be in the same cell.

accessibility relation	property	result	Z3 (s)	total (s)
R_1	Φ_1	0	–	1
	Φ_2	0.337965	59	60
R_2	Φ_1	0.337965	31	33
	Φ_2	0.337965	31	32
R_3	Φ_1	0	0	5
	Φ_2	0.621660	230	235

Table 4: Results for different amount of knowledge

N	$\lvert S \rvert$	$\lvert \mathcal{T} \rvert$	variables	constraints	Z3 (s)	total (s)
2	576	3249	806	2286	15	18
3	2916	23409	3530	10194	53	70
4	9216	88209	10406	30414	643	697
5	22500	239121	67970	175026	15433	15715

Table 5: Solving time for $Pr^{\min}[\mathbf{F}(\mathit{picture} \wedge \mathit{water})]$

size	property	result	Z3 (s)	total (s)
2	Φ_3	1.0	7	11
	Φ_4	1.0	7	11
3	Φ_3	1.0	238	258
	Φ_4	1.0	239	260
4	Φ_3	1.0	4363	4454
	Φ_4	1.0	4379	4473

Table 6: Results for Φ_3 and Φ_4

5.2 Reconnaissance Model

In this model, there is a $N \times N$ grid with 2 robots, a base for each robot, some hazard areas, and some objects in different locations. The main task of the robots is to reach an object and study its properties; to this aim, one robot has sensors to detect water while the other has a camera to take pictures. The robots' movements are deterministic while taking a picture or sampling for water are probabilistic: with positive probability it is possible to get blurred pictures or not recognizing water. Moreover, if a robot runs into a hazard or it is on an adjacent square, then it has a positive probability of damaging its sensor, which increases sharply the probability of sensor failure. The sensor can be repaired at the robot's base.

Initially, the two robots are in different cells and they know whether they are in the same location.

Table 5 shows the results regarding the minimum probability of taking a good picture and identifying water on different sizes of the grid. The resulting probability is 0, induced by a scheduler never making the robots use their sensors. Similarly to the navigation model cases, Z3 is the bottleneck of our implementation, that is anyway scaling reasonably well.

Regarding the effect of analyzing epistemic operators, consider the formulas $\Phi_3 = Pr^{\max}[\mathbf{G}(\mathit{rw\_x} \neq \mathit{rc\_x} \vee \mathit{rw\_y} \neq \mathit{rc\_y})]$ and $\Phi_4 = Pr^{\max}[\mathbf{G}\mathbf{E}_{\mathit{rw},\mathit{rc}}(\mathit{rw\_x} \neq \mathit{rc\_x} \vee \mathit{rw\_y} \neq \mathit{rc\_y})]$ which ask for the maximal probability of the two robots always (knowing of) being on different cells, respectively; the results are shown in Table 6, with the same number of variables and constraints as in Table 5. As we can see, since the robots know whether they are in the same cell, the two formulas are essentially equivalent and they require roughly the same time to be computed.

6 Conclusion and Future Work

In this paper we have considered the problem of model checking probabilistic multiagent systems with respect to a probabilistic epistemic logic. We have shown that the problem is in general undecidable; we have also shown that it becomes decidable when restricted to the class of uniform memoryless schedulers. For the latter class we have proposed a decision algorithm, analyzed its complexity, and evaluated a prototypical implementation on models from the IPPC competitions. The experimental evaluation confirms the theoretical results about the complexity of the model checking problem. As future work, we plan to investigate how to make the algorithm more scalable in practice as well as how to enrich the logic, such as by adding reward predicates or ω-regular properties.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants Nos. 61532019, 61650410658, 61761136011), by the CDZ project CAP (GZ 1023), and by the CAS/SAFEA International Partnership Program for Creative Research Teams. Yuan Feng was also partially supported by the Australian Research Council (Grant Nos. DP160101652 and DP180100691).

References

[Alur *et al.*, 2002] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[Baier and Katoen, 2008] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[de Moura and Bjørner, 2008] Leonardo de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340, 2008.

[Delgado and Benevides, 2009] Carla A. D. M. Delgado and Mario R. F. Benevides. Verification of epistemic properties in probabilistic multi-agent systems. In *MATES*, volume 5774 of *LNCS*, pages 16–28, 2009.

[Fagin *et al.*, 2004] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. MIT press, 2004.

[Hahn *et al.*, 2014] Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. IscasMC: A web-based probabilistic model checker. In *FM*, volume 8442 of *LNCS*, pages 312–317. Springer, 2014.

[Huang and Luo, 2013] Xiaowei Huang and Cheng Luo. A logic of probabilistic knowledge and strategy. In *AAMAS*, pages 845–852, 2013.

[Huang and van der Meyden, 2014] Xiaowei Huang and Ron van der Meyden. Symbolic model checking epistemic strategy logic. In *AAAI*, pages 1426–1432, 2014.

[Huang *et al.*, 2011] Xiaowei Huang, Cheng Luo, and Ron van der Meyden. Symbolic model checking of probabilistic knowledge. In *TARK*, pages 177–186, 2011.

[Huang *et al.*, 2012] Xiaowei Huang, Kaile Su, and Chenyi Zhang. Probabilistic alternating-time temporal logic of incomplete information and synchronous perfect recall. In *AAAI*, 2012.

[Jamroga and van der Hoek, 2004] Wojciech Jamroga and Wiebe van der Hoek. Agents that know how to play. *Fundam. Inform.*, 63(2-3):185–219, 2004.

[Jamroga, 2003] Wojciech Jamroga. Some remarks on alternating temporal epistemic logic. In *FAMAS*, pages 133–140, 2003.

[Jonsson and Larsen, 1991] Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277, 1991.

[Kazmierczak *et al.*, 2014] Piotr Kazmierczak, Thomas Ågotnes, and Wojciech Jamroga. Multi-agency is coordination and (limited) communication. In *PRIMA*, volume 8861 of *LNCS*, pages 91–106, 2014.

[Kwiatkowska *et al.*, 2011] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.

[Lomuscio *et al.*, 2017] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.

[Nair *et al.*, 2005] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, pages 133–139, 2005.

[Paz, 1971] Azaria Paz. *Introduction to probabilistic automata (Computer science and applied mathematics)*. Academic Press, 1971.

[Pynadath and Tambe, 2002] David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *J. Artif. Intell. Res.*, 16:389–423, 2002.

[Rabin, 1963] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[Schnoor, 2010] Henning Schnoor. Strategic planning for probabilistic games with incomplete information. In *AAMAS*, pages 1057–1064, 2010.

[Seuken and Zilberstein, 2008] Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[Sultan *et al.*, 2014] Khalid Sultan, Jamal Bentahar, Wei Wan, and Faisal Al-Saqqar. Modeling and verifying probabilistic multi-agent systems using knowledge and social commitments. *Expert Systems with Applications*, 41(14):6291 – 6304, 2014.

[van der Hoek and Wooldridge, 2002] Wiebe van der Hoek and Michael Wooldridge. Tractable multiagent planning for epistemic goals. In *AAMAS*, pages 1167–1174, 2002.

[Wan *et al.*, 2013] Wei Wan, Jamal Bentahar, and Abdessamad Ben Hamza. Model checking epistemic probabilistic logic using probabilistic interpreted systems. *Knowledge-Based Systems*, 50:279 – 295, 2013.