Parameterised Verification of Strategic Properties in Probabilistic Multi-Agent Systems

Alessio Lomuscio
Imperial College London
London, United Kingdom
a.lomuscio@imperial.ac.uk

Edoardo Pirovano
Imperial College London
London, United Kingdom
e.pirovano17@imperial.ac.uk

ABSTRACT

We present a framework for verifying strategic behaviour in an unbounded multi-agent system. We introduce a novel probabilistic semantics for parameterised multi-agent systems and define the corresponding verification problem against two probabilistic variants of alternating-time temporal logic. We define a verification procedure using an abstract model construction. We show that the procedure is complete for one variant of our specification language, and partial for the other. We present an implementation and report experimental results.

ACM Reference Format:
Alessio Lomuscio and Edoardo Pirovano. 2020. Parameterised Verification of Strategic Properties in Probabilistic Multi-Agent Systems. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

A key difficulty in deploying multi-agent systems (MAS) in critical applications is to ensure that the MAS under development meets its intended specifications. Languages such as epistemic logic and alternating-time logic (ATL) have been put forward as specification languages to reason about the complex interactions that MAS may generate. Two features of ATL [1] make it attractive for reasoning about MAS. The first is its expressiveness to encode strategic interplay among agents; the second is that its verification problem is in PTIME when complete information is assumed. Extensions of ATL such as SL[SG] have also been proposed which strictly increase the expressivity of the logic whilst preserving tractability of the model checking problem [7].

One limiting aspect of these logics is the lack of support for representing the stochastic behaviour of the agents in the system, as is done in probabilistic model checking approaches [12, 15, 19, 33]. Proposals to address this in the context of ATL have recently been put forward, as we discuss below, probabilistic variants of ATL have been introduced [13, 16]. A limitation of these approaches is that the number of agents constituting the MAS has to be known at design-time.

This is an unrealistic assumption in many scenarios, however. For example, drone swarms can be deployed in varying numbers depending on the scale of the application. In these cases, it is desirable not only to check systems of a fixed size, but to be able to give guarantees that a certain specification holds in systems of any size. The area of parameterised verification [10] is concerned with developing methods to achieve this. While in its most general formulation, the parameterised verification problem is known to be undecidable [4], restrictions to the problem have been used to identify decidable fragments [3, 17, 21]. However, no approach exists for verifying unbounded MAS against strategic properties in a stochastic setting.

In this paper we overcome this significant restriction in the state-of-the-art. Specifically, we develop a parameterised verification method, based on counter abstraction [29], for checking unbounded probabilistic MAS against strategic properties expressed in a fragment of PATL* where we do not allow nesting of strategic operators. Although for ease of presentation we consider systems composed of identical agents, the results presented here also extend to heterogeneous systems. In addition to the theoretical results that we study, the implementation that we report shows its potential usefulness when studying MAS protocols, including security inspired ones, such as the jamming scenario [34] here studied.

The rest of this paper is organised as follows. After a discussion of related work below, in Section 2 we introduce some background notions on probabilistic model checking. We introduce a novel semantics for reasoning about strategic properties in probabilistic MAS of a possibly unbounded size in Section 3. In Section 4 we introduce a model checking procedure for the parameterised verification problem. This procedure is partial in general, but complete for one variant of our specification logic. We present an implementation in Section 5 and give experimental results. We conclude in Section 6.

Related Work. Approaches to probabilistic model checking and synthesis against specifications addressing the strategic interplay of agents have been put forward [2, 6, 13, 16]. Further, implementations such as PRISM-games have been developed, thereby enabling the verification of concrete systems [11, 25]. However, in this work the number of agents in the system is constant and fixed at design-time. This is not adequate for the verification of unbounded multi-agent systems such as drone swarms, where the number of agents that make up the system at run-time may vary. This is the problem we address in this paper.

Recently, a method for the parameterised verification of strategic properties expressed in ATL has been proposed [20]. However, the systems considered there do not incorporate probabilistic aspects, as we do here. Robotic scenarios, particularly swarm robotics, often require a stochastic analysis.

Work has also been carried out on the parameterised verification of probabilistic processes [8, 9]. Our work differs from this in two ways. Firstly, the communication patterns between our agents are distinct and tailored more towards modelling AI systems than

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

network protocols. Secondly, the strategic properties we consider are more expressive than the reachability ones considered there.

Closest to the work here reported is the line of work in [27, 28], which addresses the verification of unbounded MAS against two variants of probabilistic temporal logic. However, strategic properties have not been considered in this line of work either. So, the method here presented is considerably more expressive and powerful than the above. Additionally, the models used in [28] are very distinct from ours, in that the agents act asynchronously, with synchronisation depending on the type of action performed. In the models presented here, however, the agents act in a fully synchronous manner. This is somewhat closer to the semantics presented in [27]. However, the models presented there cannot support strategic properties as the agents' choices are purely stochastic and non-determinism is not supported. Additionally, the environment used there is purely deterministic, whereas we model probabilistic environments thereby enabling for more considerably more realistic scenarios. These significant technical differences require novel verification methods, which we propose here. Further, they require a novel implementation based on a model checker that supports the verification of strategic properties, which we also provide here.

2 BACKGROUND

In this section we introduce the probabilistic model checking background and notation used throughout the paper.

Discrete Time Markov Chains. We briefly summarise *discrete time Markov chains* (DTMCs). For more background on model checking DTMCs, see [5, 18, 23].

As is typical, given a countable set A, we will use $Dist(A)$ to denote the set of probability distributions on A, i.e., functions $f : A \to [0, 1]$ such that $\sum_{a \in A} f(a) = 1$. When we have a function $t : X \to Dist(A)$, we will abbreviate $(t(x))(a)$ as $t(a|x)$.

Definition 2.1 (DTMC). A discrete-time Markov chain (DTMC) is a tuple $\mathcal{D} = \langle S, \iota, t, L \rangle$, where S is a countable set of states, $\iota \in S$ is a distinguished initial state, $t : S \to Dist(S)$ is a transition probability function, and $L : S \to 2^{AP}$ is a labelling function on a set AP of atomic propositions.

A *path* in a DTMC is a sequence of states $s_0 s_1 s_2 \ldots$, such that for every $i \in \mathbb{N}$ it is the case that $t(s_{i+1}|s_i) > 0$. We use $FPath_{\mathcal{D}}$ and $IPath_{\mathcal{D}}$ respectively, to denote the set of all finite and infinite paths starting from the initial state ι. Given a path $\omega \in (FPath_{\mathcal{D}} \cup IPath_{\mathcal{D}})$, we will use ω_i to denote the i-th state in ω and $\omega(i)$ to denote the suffix of ω obtained by removing the first i states.

For a finite path we define its probability by $\mathbf{P}_{\mathcal{D}}(s_0 \ldots s_n) \triangleq \prod_{i=0}^{n-1} t(s_{i+1}|s_i)$. This can be extended to a probability measure on sets of infinite paths, see [18, 23] for details.

Markov Decision Processes. We now introduce *Markov decision processes* (MDPs). We refer to [5, 31] for more details. We mostly follow the notation from [14].

Definition 2.2 (MDP). A Markov decision process (MDP) is a tuple $\mathcal{M} = \langle S, \iota, A, P, t, L \rangle$ where S is a finite set of states, $\iota \in S$ is a distinguished initial state, A is a finite set of actions, $P : S \to 2^A$ is a protocol function (such that $P(s) \neq \emptyset$ for all $s \in S$), $t : S \times A \to Dist(S)$ is a transition function and $L : S \to 2^{AP}$ is a labelling function on a set AP of atomic propositions.

Intuitively, a transition from a state s in an MDP occurs by first non-deterministically selecting some action $a \in P(s)$ and then transitioning to state s' with probability $t(s'|s, a)$. MDPs thus give a way of describing systems that include both probabilistic and non-deterministic choice, unlike DTMCs which do not capture the latter.

A *path* in an MDP is a sequence of states and actions $s_0 a_0 s_1 a_2 \ldots$ such that for every $i \in \mathbb{N}$ it is the case that $a_i \in P(s_i)$ and $t(s_{i+1}|s_i, a_i) > 0$. We use $FPath_{\mathcal{M}}$ ($IPath_{\mathcal{M}}$, respectively) to denote the set of all finite (infinite, respectively) paths starting from the initial state ι. For a finite path $\omega = s_0 a_0 \ldots s_n$, $last(\omega) \triangleq s_n$ denotes its last state.

In order to reason about the probability of a path occurring in an MDP, we need a way to resolve the inherent non-determinism. This is captured by a scheduler (also referred to as an *adversary*, *strategy* or *policy* in the literature), which gives the probability of each action being chosen after a certain sequence of states.

Definition 2.3 (Scheduler). Given an MDP $\mathcal{M} = \langle S, \iota, A, P, t, L \rangle$, a scheduler for $\mathcal{M}$ is a function $\sigma : FPath_{\mathcal{M}} \to Dist(A)$, such that for any finite path $\omega \in FPath_{\mathcal{M}}$, we have $\sigma(a|\omega) > 0$ only if $a \in P(last(\omega))$.

We denote by $Sch_{\mathcal{M}}$ the set of all schedulers for $\mathcal{M}$. Various classes of schedulers may be defined [14]. Note that when maximising or minimising the probability of reaching a target set of states, it is sufficient to use schedulers that are memoryless (functions that only depend on the last state of the path) and deterministic (functions onto distributions taking values only in $\{0, 1\}$, i.e. Dirac distributions).

We now proceed to define the DTMC induced by a scheduler on an MDP. This describes the purely probabilistic system that results from fixing a given choice of scheduler in an MDP.

Definition 2.4 (Induced DTMC). Given an MDP $\mathcal{M} = \langle S, \iota, A, P, t, L \rangle$ and a scheduler $\sigma : FPath_{\mathcal{M}} \to Dist(A)$, the induced DTMC $\mathcal{M}_\sigma = \langle FPath_{\mathcal{M}}, \iota, t', L' \rangle$ is defined by:

- $t' : FPath_{\mathcal{M}} \to Dist(FPath_{\mathcal{M}})$ is given by:

$$t'(\rho'|\rho) \triangleq \begin{cases} \sigma(a|\rho) \times t(s|last(\rho), a) & \text{if } \rho' = \rho a s \\ 0 & \text{otherwise} \end{cases}$$

- $L'(\rho) \triangleq L(last(\rho))$ for all $\rho \in FPath_{\mathcal{M}}$

Notice that while in general the induced DTMC might have an infinite number of states, when considering memoryless schedulers it is possible to derive an equivalent DTMC that has only finitely many states (we omit the details of this construction, which can be found in [14]). Since memoryless schedulers are sufficient when considering maximising or minimising the satisfaction of temporal properties [14], the induced DTMCs we consider later in this paper can all be regarded as finite.

3 PROBABILISTIC MULTI-AGENT SYSTEMS

In this section we introduce a semantics for describing probabilistic MAS of possibly unbounded size. We then introduce a logic for reasoning about strategic properties of these.

Semantics. Our semantics is based on a modification of [27]. The semantics given in [27] does not support the modelling of strategic behaviour since the choices of actions performed by agents are based on a fixed probability distribution. We here augment their semantics with non-determinism to allow us to reason about strategies.

To encode arbitrarily many agents with stochastic behaviour, we introduce a *probabilistic agent template* to express the behaviour of the agents in the system and an *environment* to capture the rest of the state of the system. Any concrete system is composed of a finite number of agents, instantiated from the agent template, interacting with the environment.

Definition 3.1. (Probabilistic Agent Template) A *probabilistic agent template* is a tuple $T = \langle S, \iota, Act, P, t\rangle$ where:

- The finite set $S \neq \emptyset$ represents the agent's local states.
- $\iota \in S$ is a distinguished initial state.
- $Act \neq \emptyset$ is a finite set of possible local actions, where we assume a null action $\varepsilon \in Act$ exists.
- The agent's protocol function $P : S \longrightarrow 2^{Act}$ gives the set of possible actions in each state. Note we assume that for all $s \in S$ we have $\varepsilon \in P(s)$, i.e., the null action is always possible.
- The agent's transition function $t : S \times Act_E \times 2^{Act} \times Act \longrightarrow Dist(S)$ returns a distribution on the agent's next state given its current state, the environment's action, the set of actions performed by all the agents (including the one performed by the agent being considered) and the action performed by this agent at this time-step. We assume that, for all s, X and a_E, it is the case that:
$$t(s|s, a_E, X, \varepsilon) = 1 \tag{1}$$
We also assume that, for all s', s, X, a_E and a:
$$t(s'|s, a_E, X, a) = t(s'|s, a_E, X \cup \{\varepsilon\}, a) \tag{2}$$

Notice that by (1) it is the case that agents performing the null action never change state. Further, by (2), other agents cannot observe that the null action has been performed. These two conditions ensure that agents can always choose to behave as if they are not there, and not affect the other agents.

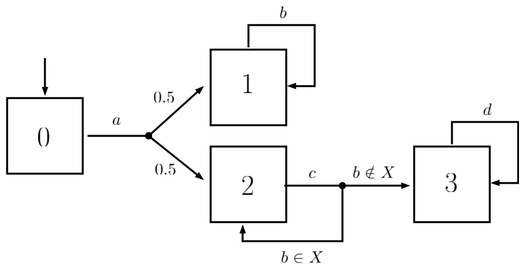
Further, note that while for ease of presentation we assume that all agents are based on the same template, the framework can be extended to accommodate a finite number of different templates. Additionally, while all agents share the same set of states and possible actions, they can each exhibit different behaviours by making different choices of actions.

We have also assumed that there is a unique initial state, rather than a probability distribution on the initial states as is sometimes done in probabilistic model checking literature. This does not reduce the expressiveness of our model, however, since the first action of the agent can be used to stochastically choose what state to begin from.

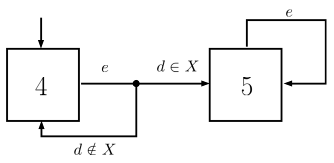
We now define the environment the agents interact with.

Definition 3.2. (Probabilistic Environment) A probabilistic environment is a tuple $E = \langle S_E, \iota_E, Act_E, P_E, t_E\rangle$ where:

- The finite set $S_E \neq \emptyset$ represents the environment's local states.
- $\iota_E \in S_E$ is a distinguished initial state.
- $Act_E \neq \emptyset$ is a finite set of possible environment actions.
- The environment's protocol function $P_E : S_E \longrightarrow 2^{Act_E}$ gives the set of possible actions in each state.



(a) An example agent template.



(b) An example environment.

Figure 1: An example probabilistic multi-agent system. We use X to denote the set of actions performed by all the agents. Note that for clarity the null actions ε are omitted.

- The environment's transition function $t_E : S_E \times 2^{Act} \times Act_E \longrightarrow Dist(S_E)$ returns a distribution on the environment's next state given its current state, the actions performed by all the agents and the action it performed.
We assume that, for all s'_E, s_E, X and a_E:
$$t_E(s'_E|s_E, X, a_E) = t(s'_E|s_E, X \cup \{\varepsilon\}, a_E) \tag{3}$$

Note that, by (3), the environment cannot observe whether a null action was performed by any of the agents, thus agents performing null actions do not affect the transition of the system in any way. Having defined its key components, we now define a probabilistic MAS as consisting of an agent template and environment, together with a labelling function.

Definition 3.3. (Probabilistic Multi-Agent System) A *probabilistic multi-agent system* (PMAS) is a tuple $\mathcal{S} = \langle T, E, \mathcal{V}\rangle$, where T is a probabilistic agent template, E is an environment and $\mathcal{V} : S \times S_E \longrightarrow 2^{AP}$ is a labelling function on a set of atomic propositions AP.

An example PMAS can be seen in Figure 1. Here, agents in the initial state 0 can perform an a action, which with equal probability takes them either to state 1 or state 2. In state 1, agents can perform the b action that does not change their state. In state 2, agents can perform a c action which takes them to state 3 but only if no agents performed the b action at the same time. Once in state 3, agents can perform the d action. The environment only has one action e that takes it to state 5 if an agent performed the d action.

A PMAS $\mathcal{S}$ gives a description of an infinite number of concrete systems that can be obtained by fixing a number n of agents in it. We now define how to obtain such a concrete model, which will be encoded as a Markov decision process (MDP) [31]. We use $\dot{n} \triangleq \{1, \ldots, n\}$ to denote the set of concrete agents.

Definition 3.4. (Concrete Model) Given a PMAS $\mathcal{S}$, a *concrete model* of n agents for $\mathcal{S}$ is an MDP $\mathcal{S}(n) = \langle G_n, \iota_n, Act_n, P_n, L_n\rangle$,

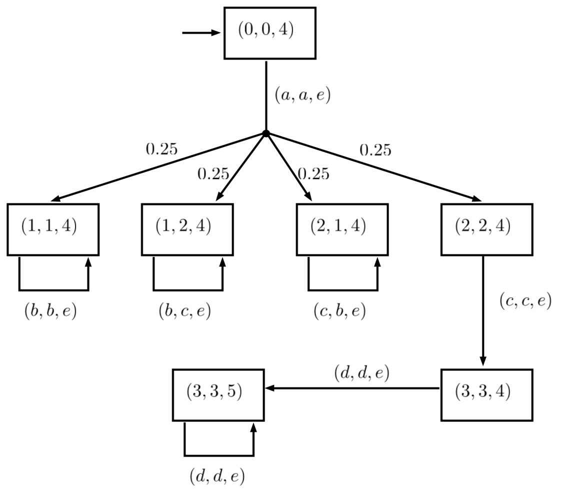


Figure 2: The concrete system corresponding to instantiating the PMAS in Figure 1 with two agents. Note that transitions involving the null action ε are omitted for clarity – the full system with null transitions has more choices of transitions and reachable states.

representing the behaviour of a global system composed of n agents and the environment, where:

- The set of *global states* $G_n = S \times \cdots \times S \times S_E$ is the set of $(n+1)$-tuples giving a local state for each of the n agents and the environment. Given a global state g we write $g.i$ to denote the local state of agent i and $g.E$ to denote the local state of the environment.
- The global state $\iota_n = (\iota, \ldots, \iota, \iota_E) \in G_n$ is the initial global state of the model $\mathcal{S}(n)$.
- The set $Act_n = Act \times \cdots \times Act \times Act_E$ is the set of global actions, i.e., $(n+1)$-tuples representing the actions for each of the n agents in $\mathcal{S}(n)$ and the environment. Given a global action a, we write $a.i$ to denote the action of agent i, and $a.E$ to denote the action of the environment.
- The protocol function $P_n : G_n \to 2^{Act_n}$ is defined by:

$$P_n(g) \triangleq \{a \in Act_n \mid \forall i \in \dot{n} : a.i \in P(g.i),\\ a.E \in P_E(g.E)\}$$

- The transition probability $t_n : G_n \times Act_n \to Dist(G_n)$ is given by:

$$t_n(g'|g, a) \triangleq t_E(g'.E|g.E, X, a.E) \times \prod_{i=1}^{n} t(g'.i|g.i, a.E, X, a.i)$$

  where $X \triangleq \{a.1, \ldots, a.n\}$ is the set of actions performed by at least one agent.
- The labelling function $L_n : G_n \to 2^{AP \times \dot{n}}$ is defined by:

$$L_n(g) \triangleq \{(p, i) \in AP \times \dot{n} \mid p \in \mathcal{V}(g.i, g.E)\}$$

An example of a concrete system can be seen in Figure 2. In the initial state, both agents can perform a and the environment can perform e. In the following states, agents in state 1 perform b and agents in state 2 perform c. If both perform c, then they can move to state 3 and perform d, causing the environment state to change to 5. Note that if the null actions were not omitted, the agents in state 1 could choose to perform the null action ε instead of b in order to allow agents in state 2 to move to state 3, so the environment state 5 would be reachable in more cases.

Notice that before we can reason about the probability of a path occurring in this concrete system we need to resolve the non-determinism of the choice of actions by the agents and environment. We do this with the definition below.

Definition 3.5. (Agent Strategy) Given a concrete system $\mathcal{S}(n)$, a strategy for an agent $i \in \dot{n}$ is a function $\sigma_i : FPath_{G_n} \to Dist(Act)$ such that for all $\omega \in FPath_{G_n}$ it is the case that $\sigma(a|\omega) > 0$ implies $a \in P(last(\omega).i)$.

A strategy for the environment σ_E is similarly defined. Given a set of agents, possibly including the environment, $A \subseteq \dot{n} \cup \{E\}$ we use σ_A to denote a strategy profile giving a strategy for each of these, i.e. $\sigma_A : A \to (FPath_{G_n} \to Dist(Act))$ with $\sigma_A(a)$ a valid strategy for every $a \in A$. We use $A^c \triangleq (\dot{n} \cup \{E\}) \setminus A$ to denote the complement of A.

When we fix a joint strategy σ for all the agents and the environment then the non-determinism in the system is eliminated and the system becomes a discrete time Markov chain (DTMC) [18] as described in Definition 2.4.

Specifications. We consider specifications based on a fragment of PATL* [13], which we call P[ATL*]. Given a set AP of atomic propositions, P[ATL*] formulae are defined by the following grammar:

$$\begin{aligned} \phi &::= \langle\langle A \rangle\rangle P_{\bowtie r}[\psi] \\ \psi &::= \top \mid (p, i) \mid \neg\psi \mid \psi \wedge \psi \mid X\psi \mid \psi U \psi, \end{aligned}$$

where $A \subset \mathbb{Z}^+ \cup \{E\}$ is a finite set of agents (and possibly the environment), $p \in AP$, $i \in \mathbb{Z}^+$, $\bowtie \in \{<, \leq, \geq, >\}$ and $r \in [0, 1]$.

Notice this corresponds to the fragment of PATL* in which we only allow one strategy operator at the top of the formula. While not as general as full PATL*, this fragment is still expressive enough to allow us to verify properties of interest in multi-agent systems, as we will see in Section 5.

The formula $\langle\langle A \rangle\rangle P_{\bowtie r}[\psi]$ is read as "agents A have a strategy to ensure that ψ occurs with probability $\bowtie r$". The temporal modality $X\psi$ means that "ψ holds at the next time-step"; $\psi_1 U \psi_2$ stands for "at some point ψ_2 holds and before then ψ_1 is true". We also use standard abbreviations such as $F\psi \equiv \top U \psi$ and $G\psi \equiv \neg F \neg \psi$.

To illustrate the language, consider an opinion formation protocol [32] where a group of robots have to agree on some choice of option. Then, the P[ATL*] formula

$$\langle\langle 2, E \rangle\rangle P_{\geq 0.5}[G\neg(decisionReached, 1)]$$

represents that agent 2 and the environment have a strategy that ensures with probability at least 0.5 that agent 1 does not reach a decision. Notice that such a property could not be expressed in previous work on verifying unbounded probabilistic systems [27, 28], which did not support expressing strategic specifications.

We say a formula is m-indexed if it refers to agents with index at most m. For instance, the example formula above is a 2-indexed formula.

We now formally define the satisfaction relation for this logic.

Definition 3.6. (Satisfaction) Given a concrete system $\mathcal{S}(n)$ and an m-indexed P[ATL*] formula $\phi = \langle\langle A\rangle\rangle P_{\bowtie r}[\psi]$ with $m \le n$, we say the formula is satisfied in $\mathcal{S}(n)$, denoted by $\phi \models \mathcal{S}(n)$ iff there is some strategy profile σ_A for the agents in A such that for all strategy profiles σ_{A^c} for the agents in A^c it is the case that:

$$\mathbf{P}_{\mathcal{S}(n)_\sigma}(\{\omega \in \mathrm{IPath}_{\mathcal{S}(n)_\sigma}(\iota) : \omega \models \psi\}) \bowtie r$$

where $\mathcal{S}(n)_\sigma$ denotes the DTMC obtained by fixing the joint strategy given by σ_A and σ_{A^c} in $\mathcal{S}(n)$. Satisfaction of path formulas in this DTMC is defined by:

$\omega \models \top$		always holds
$\omega \models (a, i)$	iff	$(a, i) \in L_n(\omega_0)$
$\omega \models \neg\psi$	iff	$\omega \not\models \psi$
$\omega \models \psi_1 \wedge \psi_2$	iff	$\omega \models \psi_1$ and $g \models \psi_2$
$\omega \models X\psi$	iff	$\omega(1) \models \psi$
$\omega \models \psi_1 U \psi_2$	iff	for some $i \ge 0$, $\omega(i) \models \psi_2$ and for all $0 \le j < i$, $\omega(j) \models \psi_1$

Notice that the set of paths defined by the path formula ψ is always measurable (see Corollary 2.4 in [33]), so the probability is well-defined.

We also consider a variant of this logic, which we call P[ATL$^*_k$], where the specifications can only describe properties of finite traces rather than infinite one. In particular, P[ATL$^*_k$] formulae are defined by the following grammar:

$$\phi' ::= \langle\langle A\rangle\rangle P_{\bowtie r}[\psi']$$
$$\psi' ::= \top \mid (p, i) \mid \neg\psi' \mid \psi' \wedge \psi' \mid X\psi' \mid \psi' U^{\le k} \psi',$$

where $k \in \mathbb{N}$ and the other components are as before.

The new operator $\psi'_1 U^{\le k} \psi'_2$ is read as "at some point within k time-steps ψ'_2 holds and before then ψ'_1 is true".

We now define the time bound of a formula. Intuitively, this encodes how many steps of a system's behaviour we need to consider in order to check the formula.

Definition 3.7. (Time Bound) The time bound $tb(\psi')$ for a P[ATL$^*_k$] path formula ψ' is defined as:

$$tb(\psi') \triangleq \begin{cases} tb(\psi'_1) & \text{if } \psi' = \neg\psi'_1 \\ max(tb(\psi'_1), tb(\psi'_2)) & \text{if } \psi' = \psi'_1 \wedge \psi'_2 \\ k + tb(\psi'_1) & \text{if } \psi' = X^k \psi'_1 \\ k + max(tb(\psi'_1), tb(\psi'_2)) & \text{if } \psi' = \psi'_1 U^{\le k} \psi'_2 \\ 0 & \text{otherwise} \end{cases}$$

For instance, in the opinion formation scenario previously considered, the P[ATL$^*_k$] formula

$$\langle\langle 2, E\rangle\rangle P_{\ge 0.5}[G^{\le 20} \neg(decisionReached, 1)]$$

represents that agent 2 and the environment have a strategy that ensures with probability at least 0.5 that agent 1 does not reach a decision *for the first 20 time-steps*. The path formula has a time-bound of 20.

Formally, the satisfaction relation for the time-bounded operator is defined by:

$\omega \models \psi'_1 U^{\le k} \psi'_2$ iff for some $0 \le i \le k$ $\omega(i) \models \psi'_2$ and for all $0 \le j < i$, $\omega(j) \models \psi'_1$

Satisfaction of the other operators is defined as before.

For the remainder of this paper, when we refer to "a formula," we mean a formula in either P[ATL*] or P[ATL$^*_k$]. Where a result applies only to one variant, this will be made explicit.

The parameterised model checking problem is concerned with checking whether a formula is satisfied in instances of any size. This is formalised below.

Definition 3.8. (PMCP) Given a PMAS $\mathcal{S}$ and an m-indexed formula ϕ, the PMCP is to determine whether $\mathcal{S}(n) \models \phi$ for all $n \ge m$. If this is the case we write $\mathcal{S} \models \phi$.

Notice that in general the PMCP is undecidable since it extends a problem that is already known to be undecidable [4] with probabilities and strategies. However, it is still of interest to explore decidable fragments of this, as we do in the next section.

4 MODEL CHECKING PROCEDURE

In this section we develop a decision procedure to the PMCP for formulas of the form $\langle\langle A\rangle\rangle P_{\ge r}[\psi]$. Other choices of inequality can be checked similarly.

The following concept will be used in the rest of the paper.

Definition 4.1. (Maximal Probability) Let $\mathcal{S}$ be a PMAS, A a coalition of agents and ψ a path formula. Then we use $\langle\langle A\rangle\rangle P_{n,max=?}[\psi]$ to denote the maximal value of $r \in [0, 1]$ for which it is the case that $\mathcal{S}(n) \models \langle\langle A\rangle\rangle P_{\ge r}[\psi]$.

Intuitively, in the above definition $\langle\langle A\rangle\rangle P_{n,max=?}[\psi]$ is the maximum probability with which the agents A can ensure ψ is achieved in a system of size n. Note that since the system of size n is finite, this is well-defined and there is a strategy that achieves it.

Observe that if we can compute the minimum and maximum values for $\langle\langle A\rangle\rangle P_{n,max=?}[\psi]$ as we range over n, we can obtain a decision procedure for the PMCP. The rest of the section is devoted to exploiting this intuition.

The following result gives the maximum value as it shows that $\langle\langle A\rangle\rangle P_{n,max=?}[\psi]$ is non-increasing.

Lemma 4.2. *Let $\mathcal{S}$ be a PMAS. Then, for any set of agents A and path formula ψ it is the case that:*

$$\langle\langle A\rangle\rangle P_{n,max=?}[\psi] \ge \langle\langle A\rangle\rangle P_{n+1,max=?}[\psi]$$

for values of n larger than the index of the formula.

Proof sketch. Notice that all agents in A appear already in the system of size n since this is larger than the index of the formula. The additional agent in the system of size $n + 1$ is thus in A^c and will not be collaborating to maximise ψ. Observe that this agent can choose a strategy where it always performs the null action, in which case the probability of ψ being satisfied that the agents in A can achieve will remain unchanged. This gives our result. □

This lemma is a probabilistic equivalent of the intuitive property that in a non-probabilistic system with null actions, adding an agent that is not part of the coalition trying to achieve a formula will not make it satisfied if it was not already.

It follows from the lemma that to compute the maximum value of $\langle\langle A\rangle\rangle P_{n,max=?}[\psi]$ as we vary n for an m-indexed formula, it suffices to compute the value of $\langle\langle A\rangle\rangle P_{m,max=?}[\psi]$. Note that it is immediate that this maximum is attained, since the system of size m achieves it.

We now develop a method to compute the minimum of this value. We do this by creating an abstract model whose states have two components: the first captures the state of the first m agents (i.e., the agents $1, \ldots,$ m which are referred to in the formula), the second records the set of states that arbitrarily many other agents are in. Note that the number of agents in each state is not recorded.

This abstract model is inspired by the counter abstraction [29] models used in [27]. However, these are adapted to the different semantics developed here.

We formalise the abstract model below.

Definition 4.3. (Abstract Model) Given a PMAS $\mathcal{S}$, an *abstract model* of m agents for the system $\mathcal{S}$ is an MDP $\bar{\mathcal{S}}(m) = \langle \bar{G}_m, \bar{\iota}_m, \bar{Act}_m, \bar{P}_m, \bar{L}_m \rangle$, where:

- The set of possible global states $\bar{G}_m \subseteq G_m \times 2^S$ have two components. The first records the state of the first m agents, the second is a set recording all local states of all other agents.
- The initial state is $\bar{\iota} \triangleq (\iota, \{\iota\})$.
- The set of possible actions is $\bar{Act}_m \triangleq Act_m \times 2^{Act}$.
- The protocol function $\bar{P}_m : \bar{G}_m \to 2^{\bar{Act}_m}$ is defined by:
$$\bar{P}_m(g, X) \triangleq \{(a, Y) \in \bar{Act}_m \mid a \in P_m(g), \\ \forall a' \in Y \exists s \in X : a' \in P(s)\}$$
- The transition probability $\bar{t}_m : \bar{G}_m \times \bar{Act}_m \to Dist(\bar{G}_m)$ is given by:
$$\bar{t}_n((g', X')|(g, X), (a, Y)) \triangleq$$
$$t_E(g'.E|g.E, Z, a.E) \times \prod_{i=1}^{m} t(g'.i|g.i, a.E, Z, a.i)$$
$$\times \begin{cases} 1 \text{ if } X' = X \cup \{s' \in S \mid \exists s \in X \exists a' \in Y \cap P(s) : \\ \qquad t(s'|s, a.E, Z, a') > 0\} \\ 0 \text{ otherwise} \end{cases}$$
where $Z \triangleq \{a.1, \ldots, a.m\} \cup Y$ is the set of actions performed by at least one agent in either the first component or the second.
- The labelling function $\bar{L}_m : \bar{G}_m \to 2^{AP \times \dot{m}}$ is given by:
$$\bar{L}_m(g, X) \triangleq L_m(g)$$
The function $\bar{L}_m$ discards information not pertaining to the first m agents, since this is not needed to evaluate an m-indexed formula.

Intuitively, the second component of the global state may grow as the system evolves from the initial state to capture all the states that it is possible for one or more of the agents to have reached.

Part of an example abstract model can be seen in Figure 3. In the initial state, the first two agents can perform action a and the environment can perform action e; their state is updated as it would be in the concrete model. The remaining agents, encoded in the second component of the state, can either do nothing or also perform a. If they perform a then the abstract model assumes that each of states 1 and 2 is reached by at least one of them, and the second component is updated to $\{0, 1, 2\}$.

Notice that when defining a joint strategy for the abstract model, this also includes a choice of the set of actions performed by the agents in the second component of the state. As in Definition 4.1,

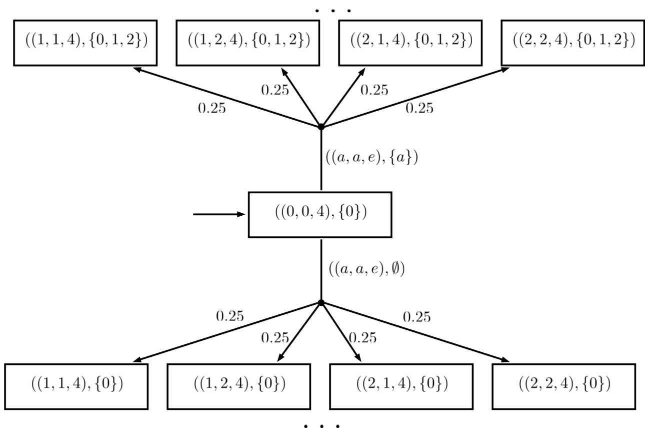


Figure 3: The initial state and first set of transitions for the abstract model with two agents of the PMAS in Figure 1. We omit the null actions ε for clarity.

we can define the maximal probability in the abstract model of m agents. We denote this by $\langle\langle A \rangle\rangle \bar{P}_{m,max=?}[\psi]$.

This brings us to the other main result of this section, which gives a lower bound on the values of $\langle\langle A \rangle\rangle P_{n,max=?}[\psi]$.

Lemma 4.4. *Let $\mathcal{S}$ be a PMAS. Then, for any set of agents A and path formula ψ it is the case that:*
$$\langle\langle A \rangle\rangle P_{n,max=?}[\psi] \geq \langle\langle A \rangle\rangle \bar{P}_{m,max=?}[\psi]$$
where m is the index of the formula and $n \geq m$.

Proof sketch. Let $\bar{\sigma}_A$ denote the strategy for the agents A in $\bar{\mathcal{S}}(m)$ that achieves the maximum probability $\langle\langle A \rangle\rangle \bar{P}_{m,max=?}[\psi]$. Consider the strategy σ_A in $\mathcal{S}(n)$ which behaves in the same way. Now notice that this strategy also achieves at least the same probability since the agents in the second component of $\bar{\mathcal{S}}(m)$ can choose any actions that the extra agents in $\mathcal{S}(n)$ can. This shows our desired result. □

Intuitively, the abstract model represents a more powerful opponent for the agents in A than any concrete system, since within the second component it captures the possible behaviours of an arbitrarily large number of agents.

When considering only formulas from the restricted P[ATL$^*_k$] logic, we also have that the probability given by the abstract model is arbitrarily approached by sufficiently large systems. This is formalised below.

Lemma 4.5. *Let $\mathcal{S}$ be a PMAS. Then, for any set of agents A, P[ATL$^*_k$] path formula ψ' and $\varepsilon > 0$ it is the case that there is some $n \in \mathbb{Z}^+$ such that:*
$$\langle\langle A \rangle\rangle P_{n,max=?}[\psi'] < \langle\langle A \rangle\rangle \bar{P}_{m,max=?}[\psi'] + \varepsilon$$
where m is the index of the formula.

Proof sketch. Notice that we are only interested in the first $tb(\psi')$ steps of behaviour of the system. Given a number of agents k, we denote by P_k the maximum probability that the agents can achieve of reaching every reachable state at every transition for the first $tb(\psi')$ time-steps. Notice that P_k tends to 1 as k increases,

Algorithm 1 Decision Procedure for the PMCP

```
Input: PMAS $\mathcal{S}$, $m$-indexed formula $\phi = \langle\langle A\rangle\rangle P_{\geq r}[\psi]$
Output: Boolean
1: if $r \leq$ MaximalProb $(\bar{\mathcal{S}}(m), A, \psi)$ then
2:     return true
3: else if $\phi \in$ P[ATL$_k^*$] then
4:     return false
5: end if
6: for $i \leftarrow m, m+1, m+2, \ldots$ do
7:     if MaximalProb $(\mathcal{S}(i), A, \psi) < r$ then
8:         return false
9:     end if
10: end for
```

since if there are many agents there is a high chance at least one will follow each of the finitely many transition that can occur in the first $tb(\psi')$ time-steps. Further, when every reachable state is reached at every transition, the path in the concrete system has a corresponding path in the abstract model. Thus, it must be the case that $\langle\langle A\rangle\rangle P_{k,max=?}[\psi']$ approaches $\langle\langle A\rangle\rangle \bar{P}_{m,max=?}[\psi']$ as k increases, and the result follows. □

Exactly computing a sufficiently large value of n for the result to hold can be carried out similarly to the proof of Theorem 4 in [27].

The lemma above shows that for P[ATL$_k^*$] formulas, sufficiently large systems achieve probabilities that are arbitrarily close to the lower bound computed using the abstract model, i.e. this bound is a tight one.

This leads us to our procedure, which is shown in Algorithm 1. We use MaximalProb to denote a procedure that uses existing techniques for probabilistic model checking of finite-state systems [25] to compute the maximum probability as in Definition 4.1. We now show the correctness of our algorithm.

Theorem 4.6. *For all PMAS $\mathcal{S}$ and formulas $\phi = \langle\langle A\rangle\rangle P_{\geq r}[\psi]$ for which* Algorithm1 *returns a value, it is the case that $\mathcal{S} \models \phi$ iff* Algorithm1$(\mathcal{S}, \phi)$ *= true.*

Proof sketch. Suppose a value was returned on line 2. By Lemma 4.4, we have that MaximalProb $(\bar{\mathcal{S}}(m), A, \psi)$ is a lower bound for the values of $\langle\langle A\rangle\rangle P_{n,max=?}[\psi]$ as n varies. Thus, since the required probability r is less than the lower bound, our property must hold.

If a value is returned on line 4, then the required probability r was higher than the lower bound. Further, our formula is a P[ATL$_k^*$] one, so by Lemma 4.5 we know that this lower bound is tight. Thus, the property cannot hold.

Finally, if we return on line 8, then there is some value of i for which it is the case that MaximalProb $(\mathcal{S}(i), A, \psi) < r$, giving a counterexample to our property holding. □

Notice that this procedure is complete for P[ATL$_k^*$] formulas, but for a general P[ATL*] one it may not terminate, as we may just test systems of increasing size in the loop on line 6 and never find a counterexample. Further, note that even in the complete case the procedure may take time exponential in the number states of the agent template since the abstract model that we check considers all subsets of these. Despite this theoretical intractability, we will see in the next section that our method can still be used to verify practical examples of systems.

5 IMPLEMENTATION

We implemented the procedure described in the previous section in an experimental toolkit called PSV-S (**P**robabilistic **S**warm **V**erifier for **S**trategic Properties), which is released as open-source [30]. Our tool is programmed in Java and based on an extension of PRISM-games [11] that can handle concurrent games [25]. This in turn builds on PRISM [24] by extending it to deal with strategic properties.

The tool takes as input a system file describing the behaviour of an agent and an environment, along with a properties file describing the properties that we wish to consider. It then constructs, based on its configuration, either the abstract model (according to Definition 4.3) in order to compute the lower bound on the probability according to Lemma 4.4, or the concrete model of a given size (according to Definition 3.4).

In order to verify the functionality and scalability of our tool, we used it to model a channel jamming security protocol [34]. In our model of the protocol, there are k channels available to an agent to send messages. At each time-step, an agent can choose one channel to send a message. A number of agents in the system are attackers; each of them can jam a channel. It is assumed that if a message is sent along a non-jammed channel, then it is successfully transmitted with probability 0.4. If it is sent along a channel that is jammed by at least one attacker then this probability drops to 0.1.

The sending of messages and jamming of channels is modelled in an agent template, with the choices of action for the agent at each time-step being to send along a channel i (with $0 \leq i < k$) or block a channel i (with $0 \leq i < k$). The environment acts as a receiver for the messages, and tracks how many messages have been received.

We wish to verify the property that with probability p a nominated agent (agent 1) can ensure that at least i messages are transmitted within j time-steps. This can be expressed by the PATL$_k^*$ property

$$\langle\langle 1\rangle\rangle P_{\geq p}[F^{\leq j}(transmitted_i, 1)]$$

where $transmitted_i$ is an atomic proposition that holds when the user has transmitted at least i messages.

Notice that, as observed in Section 4, in order to analyse properties of this form it is sufficient to find the value of

$$\langle\langle 1\rangle\rangle P_{n,max=?}[F^{\leq j}(transmitted_i, 1)]$$

which represents the maximum probability that can be achieved by agent 1 in a system of size n, as we vary the parameter n.

All results were obtained on a machine running OpenJDK 1.8.0 (64-bit version) and Ubuntu 18.04 (Linux kernel 4.15.0-69) with an Intel i7-7700HQ processor and 24GB of RAM (out of 32GB total) allocated to the JVM heap.

For our first experiment, we fixed the number of channels available to 4, the number of messages being transmitted to 3, and the number of time-steps allowed for transmission to 15. We then used

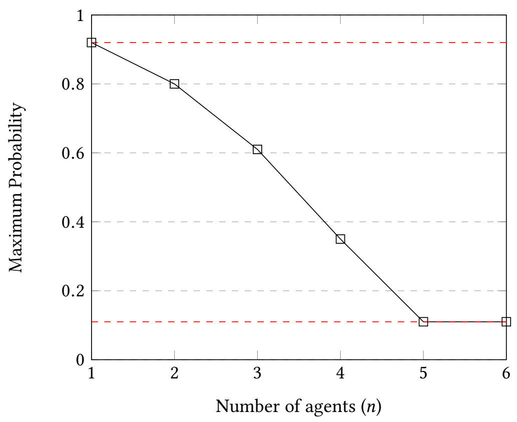


Figure 4: Graph showing the maximum probability $\langle\langle 1\rangle\rangle P_{n,max=?}[F^{\leq 15}(transmitted_3, 1)]$ for different values of n. The number of channels is fixed to 4. The red dashed lines show the expected lower and upper bounds according to Lemmas 4.2 and 4.4.

		k		
		3	4	5
i	5	0.37 / 0.28 24 / 14,976	3.80 / 2.29 24 / 78,336	86.66 / 35.89 24 / 387,072
	10	0.56 / 0.47 44 / 27,456	6.40 / 3.29 44 / 143,616	156.35 / 77.77 44 / 709,632
	25	0.98 / 0.98 104 / 64,896	15.42 / 7.30 104 / 339,456	395.20 / 157.90 104 / 1,677,312
	50	1.80 / 2.27 204 / 127,296	29.13 / 16.76 204 / 665,856	*timeout*
	75	2.52 / 3.34 304 / 189,696	41.63 / 22.12 304 / 992,256	*timeout*

Table 1: For different values of k (available channels) and i (number of messages to transmit), the time needed (in seconds) to respectively build the abstract model and use this to compute the minimum value of $\langle\langle 1\rangle\rangle P_{n,max=?}[F^{\leq 150}(transmitted_i, 1)]$, along with the number of states and transitions in this model. Timeouts indicate a time longer than 10 minutes.

our tool to compute the upper and lower bounds for

$$\langle\langle 1\rangle\rangle P_{n,max=?}[F^{\leq 15}(transmitted_3, 1)]$$

as n varies (as given by Lemma 4.2 and Lemma 4.4, respectively), as well as the actual values of this for different values of n. Our results are shown in Figure 4.

As expected, the actual values fall within the calculated range. Further, the minimum value is attained once there are at least 5 agents in the system. This is expected since this corresponds to the system where there are 4 attackers, which is enough for them to have a joint strategy to block every channel. The computation of the minimum and maximum values is much more efficient than constructing systems of increasing size. The former takes around 5 seconds, the latter takes approximately 450 seconds for checking just the system of size 5.

As a further experiment to verify the scalability of our tool, we checked the time taken to construct the abstract model and use this to compute the minimum probability of

$$\langle\langle 1\rangle\rangle P_{n,max=?}[F^{\leq 150}(transmitted_i, 1)]$$

as we varied the number of messages that we wished to receive and the number of channels. Our results are in Table 1, along with the total number of states and total number of transitions in the abstract model.

Notice that varying the number of channels changes only the number of transitions in the model since there are more choices of channel to transmit on but the number of messages we have to keep track of remains unchanged. Varying the number of messages being sent, on the other hand, also changes the number of states.

No comparison of our tool's performance to others is provided since, to the best of our knowledge, no other tool allows the verification of strategic properties in probabilistic multi-agents systems with a possibly unbounded number of agents. Further, in order to allow us to express strategic properties, the semantics have to be different from those in previous work on verifying temporal properties of unbounded probabilistic systems [27, 28]. Thus, even a comparison to the verification of a temporal property is not possible as the models we use are distinct.

We note also that if we fix a number of agents then our tool will exhibit similar timings to the PRISM-games extension it uses as its underlying model checker [25], since it will be calling the subroutines there to verify the concrete system. Since PRISM-games has already been benchmarked on many case studies [26], describing such benchmarking results here would not be of interest.

6 CONCLUSIONS

We have presented a method for verifying strategic properties in multi-agent systems that are probabilistic and unbounded in size. To the best of our knowledge, no other work addresses this problem. After introducing a novel semantics for reasoning about such systems, we gave a decision procedure for the verification problem based on constructing an abstract model using a counter abstraction technique. We showed that our decision procedure is sound, and also noted that it is complete on the variant of our specification logic describing properties of finite traces. Finally, we implemented our technique, and used our implementation to verify properties of a channel jamming scenario.

In future work, we plan to apply the method and tool described to further scenarios, such as protocols from swarm robotics. We also plan to explore further extensions to the logic, such as probabilistic variants of epistemic properties [15]. Finally, we intend to combine this work with techniques developed to verify fault tolerance of multi-agents systems [22], in order to verify strategic properties in systems that may exhibit faults.

ACKNOWLEDGEMENTS

Alessio Lomuscio is supported by a Royal Academy of Engineering Chair in Emerging Technologies. The authors would like to thank the anonymous reviewers for valuable feedback and suggestions.

REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49, 5 (2002), 672–713.

[2] B. Aminof, M. Kwiatkowska, B. Maubert, A. Murano, and S. Rubin. 2019. Probabilistic Strategy Logic. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI19)*. 32–38.

[3] B. Aminof, S. Rubin, I. Stoilkovska, J. Widder, and F. Zuleger. 2018. Parameterized Model Checking of Synchronous Distributed Algorithms by Abstraction. In *Proceedings of the 19th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI18)*. Springer International Publishing, 1–24.

[4] K.R. Apt and D. C. Kozen. 1986. Limits for automatic verification of finite-state concurrent systems. *Inform. Process. Lett.* 22, 6 (1986), 307–309.

[5] C. Baier and J. P. Katoen. 2008. *Principles of Model Checking.* The MIT Press.

[6] N. Basset, M. Kwiatkowska, U. Topcu, and C. Wiltsche. 2015. Strategy Synthesis for Stochastic Games with Multiple Long-Run Objectives. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS15) (Lecture Notes in Computer Science)*, Vol. 9035. Springer, 256–271.

[7] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, and A. Murano. 2019. Strategy Logic with Simple Goals: Tractable Reasoning about Strategies. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI19)*. 88–94.

[8] N. Bertrand and P. Fournier. 2013. Parameterized Verification of Many Identical Probabilistic Timed Processes. In *Proceedings of the 33rd Foundations of Software Technology and Theoretical Computer Science conference (FSTTCS13)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 501–513.

[9] N. Bertrand, P. Fournier, and A. Sangnier. 2014. Playing with Probabilities in Reconfigurable Broadcast Networks. In *Proceedings of the 17th International Conference on Foundations of Software Science and Computation Structures (FOSSACS14) (Lecture Notes in Computer Science)*, Vol. 8412. Springer, 134–148.

[10] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. 2015. *Decidability of Parameterized Verification.* Morgan and Claypool Publishers.

[11] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. 2013. Automatic Verification of Competitive Stochastic Systems. *Formal Methods in System Design* 43, 1 (2013), 61–92.

[12] T. Chen, V. Forejt, M. Z. Kwiatkowska, D. Parker, and A. Simaitis. 2013. PRISM-games: A Model Checker for Stochastic Multi-Player Games. In *Proceedings of the 19th International Conference for Tools and Algorithms for the Construction and Analysis of Systems (TACAS13) (Lecture Notes in Computer Science)*, Vol. 7795. Springer, 185–191.

[13] T. Chen and J. Lu. 2007. Probabilistic Alternating-time Temporal Logic and Model Checking Algorithm. In *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD07)*. IEEE Computer Society, 35–39.

[14] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. 2011. Automated verification techniques for probabilistic systems. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems (Lecture Notes in Computer Science)*, Vol. 6659. Springer, 53–113.

[15] X. Huang and C. Luo. 2013. A logic of probabilistic knowledge and strategy. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS13)*. IFAAMAS, 845–852.

[16] X. Huang, K. Su, and C. Zhang. 2012. Probabilistic Alternating-Time Temporal Logic of Incomplete Information and Synchronous Perfect Recall. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI12)*. AAAI Press, 765–771.

[17] A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. 2013. Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In *Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 201–209.

[18] J. G. Kemeny, J. Laurie Snell, and A. W. Knapp. 1976. *Denumerable Markov Chains* (2 ed.). Springer.

[19] S. Konur, C. Dixon, and M. Fisher. 2012. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems* 60, 2 (2012), 199–213.

[20] P. Kouvaros and A. Lomuscio. 2016. Parameterised Model Checking for Alternating-time Temporal Logic. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI16)*. IOS Press, 1230–1238.

[21] P. Kouvaros and A. Lomuscio. 2016. Parameterised Verification for Multi-Agent Systems. *Artificial Intelligence* 234 (2016), 152–189.

[22] P. Kouvaros, A. Lomuscio, and E. Pirovano. 2018. Symbolic Synthesis of Fault-Tolerance Ratios in Parameterised Multi-Agent Systems. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence (IJCAI-ECAI18)*. IJCAI, 324–330.

[23] M. Kwiatkowska, G. Norman, and D. Parker. 2007. Stochastic Model Checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07) (Lecture Notes in Computer Science)*, Vol. 4486. Springer, 220–270.

[24] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV11)*. Springer, 585–591.

[25] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. 2018. Automated Verification of Concurrent Stochastic Games. In *Proceedings of the 15th International Conference on Quantitative Evaluation of SysTems (QEST18) (Lecture Notes in Computer Science)*, Vol. 11024. Springer, 223–239.

[26] M. Kwiatkowska, D. Parker, and C. Wiltsche. 2018. PRISM-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives. *STTT* 20, 2 (2018), 195–210.

[27] A. Lomuscio and E. Pirovano. 2018. Verifying Emergence of Bounded Time Properties in Probabilistic Swarm Systems. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and 23rd European Conference on Artificial Intelligence (IJCAI-ECAI18)*. IJCAI, 403–409.

[28] A. Lomuscio and E. Pirovano. 2019. A Counter Abstraction Technique for the Verification of Probabilistic Swarm Systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*. ACM, 161–169.

[29] A. Pnueli, J. Xu, and L. Zuck. 2002. Liveness with (0, 1,infinity)-counter abstraction. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV02) (Lecture Notes in Computer Science)*, Vol. 2404. Springer, 93–111.

[30] PSV-S. 2020. Probabilistic Swarm Verifier for Strategic Properties http://vas.doc.ic.ac.uk/software/psv-s.html. (2020).

[31] M.L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc.

[32] G. Valentini, H. Hamann, and M. Dorigo. 2015. Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS Press, 1305–1314.

[33] M. Y. Vardi. 1985. Automatic Verification of Probabilistic Concurrent Finite-State Programs. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS85)*. IEEE, 327–338.

[34] Q. Zhu, H. Li, Z. Han, and T. Basar. 2010. A Stochastic Game Model for Jamming in Multi-Channel Cognitive Radio Systems. In *Proceedings of IEEE International Conference on Communications, (ICC10)*. IEEE, 1–6.