ECAI 2016
G.A. Kaminka et al. (Eds.)
© 2016 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/978-1-61499-672-9-1230

Parameterised Model Checking for Alternating-Time Temporal Logic

Panagiotis Kouvaros[1] and **Alessio Lomuscio**[1]

Abstract. We investigate the parameterised model checking problem for specifications expressed in alternating-time temporal logic. We introduce parameterised concurrent game structures representing infinitely many games with different number of agents. We introduce a parametric variant of ATL to express properties of the system irrespectively of the number of agents present in the system. While the parameterised model checking problem is undecidable, we define a special class of systems on which we develop a sound and complete counter abstraction technique. We illustrate the methodology here devised on the prioritised version of the train-gate-controller.

1 Introduction

In the past 15 years there has been considerable interest in logics and techniques for reasoning about strategic play in multi-agent systems (MAS). Formalisms such as Alternating-Time Logic [4] enable us to capture precisely what agents may bring about by cooperating with one another. They also offer a game-theoretic angle to study interactions within MAS that was missing in previous mainstream formalisms, e.g., in the plain BDI approach [35].

Formalisms stronger than ATL have been put forward. The logic ATL* [4] extends the cooperation modalities to the full power of CTL* temporal modalities. More recently, a wide range of so called "strategy logics" [10, 32] have been introduced to increase the expressiveness even further. For example, in a variant of strategy logic, strategies can be explicitly named and bound to agents. This is useful in many scenarios and results in being able to express, for example, concepts such as Nash equilibria. Several underlying conditions on MAS have been explored in this context, including complete and incomplete information, perfect recall and memoryless strategies.

There is, however, a whole class of MAS that has not received attention so far in terms of *strategic reasoning*. This concerns *MAS that are composed by an unbounded number of components*. This is a natural and general class of MAS not only of theoretical interest, but of considerable relevance to applications. For example, open MAS where components enter and leave the system at runtime may be formalised by MAS with unbounded numbers of agents [7, 33]. Robotic swarms are also naturally assumed to be systems in which the number of agents is unbounded [27, 28, 29]. This tradition dates back to applications such as networking where protocols are often analysed by considering the unbounded nature of the processes in the system [8].

Specifications concerning strategic interplay naturally arise in these application areas. For example, in swarm analysis the engineer may be interested in establishing whether a small coalition in a swarm can influence the behaviour of the whole system in a certain way (perhaps by forcing its direction of travel). In open MAS such as auctions it may be of interest to establish whether a small set of players can collude to force an unwanted state of affairs. Strategic interplay on games with infinitely many players is also of theoretical interest: many games are defined on arbitrarily many players including game cutting, diner's dilemma, etc. In all these scenarios it is of interest to establish whether a *strategic property holds on the game irrespective of the number of players in the system*. By establishing the above we can deduce a strong property of the system, as the property will have been guaranteed to hold irrespective of the particular system instance under analysis.

In this paper we put forward a methodology to address this problem. We assume that our MAS is composed by arbitrarily many agents. We express specifications of interest in a novel variant of ATL, where we can naturally encode whether all players of a particular kind can, by establishing a coalition, force states of affairs expressed as LTL formulas on the rest of the system. We define the verification problem in terms of the verification of all realisations of the system, and show the problem is undecidable in general. Our key contribution is a technique based on counter abstraction that enables us, in several cases of interest, to draw conclusions on all possible system instances by analysing a single abstract model.

Related Work. We are not aware of any work addressing the problem above. Of course, the verification problem for ATL has received much attention over the years, including various implementations from the seminal jMocha model checker [2] to recent symbolic implementations such as Verics [24] and MCMAS [31]. However, all these techniques deal with a finite number of agents, and so cannot address the general problem. Closer to our approach is recent work on parameterised verification for MAS [25, 26, 30], where unbounded MAS are checked against epistemic specifications. In particular, [27, 28] present counter abstraction techniques to verify unbounded MAS against epistemic specifications. While our technique is inspired by [27, 28], the specification language supported here is based on strategies, and so it is not directly comparable.

Scheme of the Paper. In Section 2 we introduce parameterised concurrent game structures (PCGS), a novel form of concurrent game structures that can represent arbitrarily many agents; this is followed by the introduction of PATL, a parameterised version of ATL. In Section 3 we prove that the parameterised model checking problem for PCGS against PATL is undecidable and identify a class of systems for which we give a method for parameterised verification in Section 4. In Section 5 we encode an unbounded version of the train-gate-controller in PCGS and express relevant specificactions in PATL. We conclude in Section 6, where we discuss possible future work.

[1] Department of Computing, Imperial College London, UK, email: {p.kouvaros,a.lomuscio}@imperial.ac.uk

2 Parameterised Concurrent Game Structures

To reason about strategic interplay in unbounded MAS, we introduce the formalism of *parameterised concurrent game structures* (PCGS). PCGS extend concurrent game structures [4], the traditional semantics for ATL, to represent an unbounded number of systems (or games) with an unbounded number of agents. The behaviour of the agents is given by templates which form part of the PCGS. We assume a finite number of templates from which an unbounded number of agents may be constructed. A vector of parameters specifies the number of agents in the system that are built from the templates. In other words, given a vector $(n_1, \ldots, n_k)$ and a PCGS of k templates, the concrete concurrent game structure is constructed by composing n_i agents for each template i.

We consider concrete structures where: (i) the agents have *incomplete information*, i.e., the agents are only aware of their private (local) state; (ii) the agents have *imperfect recall*, i.e., their strategies are given as functions from local states to actions; (iii) the agents' strategies are *uniform*, i.e, a strategy always specifies the same action in a given local state.

This is not the mainstream setting in which ATL was defined [4], but it is widely considered in the literature [1, 9, 22]. Observe, however, that assuming complete information, i.e., assuming that the agents are fully aware of the system's state, is problematic for unbounded systems. In fact, under complete information, each agent can be aware of the number of agents in each system instance. This immediately leads to undecidability of the parameterised model checking problem [17]. Similarly, observe that the plain model checking problem under incomplete information and perfect recall is undecidable [14]. It follows that the parameterised model checking problem is also undecidable. Finally, in the context of incomplete information and memoryless strategies we assume that strategies are *uniform* [23], i.e., when following a strategy the agents consistently select the same action in the same local state. This is an intuitive requirement that preserves the meaning of the ATL operators, only causing a jump to Δ_P^2 [22].

We define parameterised concurrent game structures. Then we define the concrete structures that these generate.

Definition 1 (Parameterised concurrent game structure) *A* parameterised concurrent game structure *(PCGS) is a tuple* $\hat{S} = \left\langle \hat{\Sigma}, \hat{Q}, \hat{q_0}, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \right\rangle$, *where:*

- $\hat{\Sigma} = \hat{\Sigma_T} \cup \Sigma_C$, *where* $\hat{\Sigma_T} = \left\{\hat{1}, \ldots, \hat{k}\right\}$ *is a finite, nonempty set of* agent templates *and* $\Sigma_C = \{1, \ldots, z\}$ *is a finite set of concrete agents.*
- $\hat{Q}$ *is a nonempty, and finite set of* local states *for the agents.*
- $\hat{q_0} = \hat{\Sigma} \to 2^{\hat{Q}}$ *describes a set of initial local states for each of the agents.*
- $\hat{A}$ *is a nonempty, and finite set of actions available to the agents.*
- $\hat{\Pi}$ *is a finite set of propositional variables.*
- $\hat{\pi} : \hat{Q} \to 2^{\hat{\Pi}}$ *is a labelling function on the states.*
- $\hat{d} : \hat{\Sigma} \times \hat{Q} \to 2^{\hat{A}}$ *determines the set of actions available to an agent given its local state.*
- $\hat{\delta} : \hat{\Sigma} \times \hat{Q} \times \hat{A} \times 2^{\hat{A}} \to \hat{Q}$ *gives the temporal evolution of an agent given its state, its action, and the projection of the joint action for the other agents into a set.*

As the above definition suggests, to account for the unbounded nature of parameterised games, information regarding the identities of the agents is abstracted away. In other words, as we will make precise with the definition of a concrete system, the temporal evolution of an agent depends on the agent's local state, its action, and the actions of the other agents, but it does not depend on how many and which agents performed each action.

Assume a PCGS $\hat{S}$ defining k agent templates. Let $\overline{n} \in \mathbb{N}^k$ be a vector of parameters for the system. Consider $\overline{n}.i$ to be the i-th component in $\overline{n}$. We now define the $\overline{n}$-st concrete instantiation $S(\overline{n})$ of $\hat{S}$. $S(\overline{n})$ is a concurrent game structure resulting from the composition of the concrete agents Σ_C with $\overline{n}.i$ instantiations $\{(i,1), \ldots, (i, \overline{n}.i)\}$ of each agent template $\hat{i}$. Let Σ denote the set of all concrete agents. The global states of $S(\overline{n})$ correspond to tuples of local states for all the agents in the system. For a global state q and an agent $ag \in \Sigma$, we write $q.ag$ for the local state of ag in q. The system's temporal evolution from a state depends on the joint action performed at the state. Given a joint action $\overline{a}$ and an agent $ag \in \Sigma$, we write $\overline{a}.ag$ for the action of ag in $\overline{a}$. By $Actions(\overline{a})$, we denote the set $Actions(\overline{a}) = \{\overline{a}.ag \colon ag \in \Sigma\}$ of actions for all the agents in $\overline{a}$; by $Actions_{-ag}(\overline{a})$, we denote the set $Actions_{-ag}(\overline{a}) = \{\overline{a}.ag' \colon ag' \in \Sigma \setminus \{ag\}\}$ of actions for all the agents other than ag in $\overline{a}$.

Definition 2 (Concrete concurrent game structure) *Let* $\hat{S} = \left\langle \hat{\Sigma}, \hat{Q}, \hat{q_0}, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \right\rangle$ *be a PCGS of k agent templates and z concrete agents. Let* $\overline{n} \in \mathbb{N}^k$. *A* concrete concurrent game structure *(CCGS) is a tuple* $S(\overline{n}) = \langle \Sigma, Q, Q_0, A, \Pi, \pi, d, \delta \rangle$, *where:*

- $\Sigma = \Sigma_T \cup \Sigma_C$, *where* $\Sigma_T = \{(i,j) \colon 1 \leq i \leq k, 1 \leq j \leq \overline{n}.i\}$ *is the set of concrete agents instantiated from the templates.*
- $Q = \hat{Q}^{|\Sigma|}$ *is the set of* concrete states.
- $Q_0 = \left(\hat{q_0}(\hat{1})\right)^{\overline{n}.1} \times \ldots \times \left(\hat{q_0}(\hat{k})\right)^{\overline{n}.k} \times (\hat{q_0}(1)) \times \ldots (\hat{q_0}(z))$ *is the set of initial concrete states.*
- $A = \hat{A}^{|\Sigma|}$ *is the set of joint actions.*
- $\Pi = \hat{\Pi} \times \Sigma$.
- $\pi : Q \to 2^{\Pi}$ *is defined as* $(p, ag) \in \pi(q)$ *iff* $p \in \hat{\pi}(q.ag)$.
- $d : Q \to 2^A$ *gives the set of joint actions available at a state of the system:* $\overline{a} \in d(q)$ *iff*
$$\overline{a}.i \in \hat{d}(i, q.i) \textit{ for every agent } i \in \Sigma_C, \textit{ and}$$
$$\overline{a}.(i,j) \in \hat{d}\left(\hat{i}, q.(i,j)\right) \textit{ for every agent } (i,j) \in \Sigma_T.$$
- $\delta : Q \times A \to Q$ *is the temporal transition function of the system:* $\delta(q, \overline{a}) = q'$ *iff* $\overline{a} \in d(q)$,
$$\hat{\delta}\left(q.i, \overline{a}.i, Actions_{-i}(\overline{a})\right) = q'.i \textit{ for every agent } i \in \Sigma_C, \textit{ and}$$
$$\hat{\delta}\left(q.(i,j), \overline{a}.(i,j), Actions_{-(i,j)}(\overline{a})\right) = q'.(i,j) \textit{ for every agent } (i,j) \in \Sigma_T.$$

So a PCGS generates different CCGS depending on the vector of parameters for the system. Each CCGS is composed of a different number of agents. The propositional variables in a CCGS are indexed by each of the concrete agents. A propositional variable (p, ag) holds in a global state if the agent ag is at a local state labelled with the non-indexed propositional variable p by the template labelling function. These assumptions are crucial in expressing *collective* specifications that typically accommodate games with an unbounded number of agents such as cooperative games [36] or agreement protocols [34].

A path in a CCGS is a sequence $\lambda = q_0 q_1 \ldots$ such that for every $i \geq 0$ there is a joint action $\overline{a}$ with $\delta(q_i, \overline{a}) = q_{i+1}$. For a path λ and

$i \geq 0$, we write λ^i to denote the suffix $q_i q_{i+1} \ldots$ of λ. We assume that the transition relation for CCGS is serial [2].

Alternating-time temporal logic (ATL) generalises *branching-time temporal logic* (CTL) by allowing selective quantification over paths representing possible outcomes of a system [4]. Hence, ATL specifications can express strategic profiles of agents that can enforce a certain outcome. More specifically, in addition to customary temporal modalities, ATL includes *cooperation* formulas such as $\langle\langle\Gamma\rangle\rangle\varphi$, where Γ is a group of agents, expressing that Γ has a cooperative strategy to enforce an LTL formula φ irrespective of how the other agents act.

Let $VAR = VAR_1 \cup \ldots \cup VAR_k$ be the union of disjoint sets of variable symbols, where each VAR_i is associated with agent template $\hat{i}$. To reason about unbounded groups of agents, we here define parameterised alternating-time temporal logic (PATL), a variation of ATL, where: (i) Γ is not a subset of agents from a predetermined set as in ATL, but it is a subset $\Delta \subseteq \Sigma_C \cup VAR$ of the union of the set of concrete agents Σ_C and the set of variables VAR; (ii) the atomic propositions are indexed by the variables in VAR. The domain of a variable $v \in VAR_i$ appearing in a formula φ depends on the CCGS on which φ is evaluated; if φ is evaluated on $S(\overline{n})$, then the potential set of values for v is $\{(i,1), \ldots, (i, \overline{n}.i)\}$. Intuitively, PATL formulae quantify over the concrete agents. For instance, $\forall v \langle\langle\{v\}\rangle\rangle\varphi$, where $v \in VAR_i$, expresses that every concrete agent from template $\hat{i}$ can enforce φ independently of the action performed by *however many* other agents are instantiated from each template.

Given a set Σ of concrete agents, a set $VAR = VAR_1 \cup \ldots \cup VAR_k$ of variable symbols, and a set $\hat{\Pi}$ of propositional variables, PATL formulae are defined by the following BNF grammar:

$$\varphi ::= \top \mid (p, v) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle\Delta\rangle\rangle X\varphi \mid \langle\langle\Delta\rangle\rangle(\varphi U \varphi) \mid \langle\langle\Delta\rangle\rangle G\varphi \mid \forall\varphi$$

where p is a propositional variable, v is a variable symbol, and $\Delta \subseteq \Sigma \cup VAR$.

We abbreviate $\langle\langle\Delta\rangle\rangle(\top U \varphi)$ as $\langle\langle\Delta\rangle\rangle F\varphi$. The formula $\langle\langle\Delta\rangle\rangle X\varphi$ is read as "$\langle\langle\Delta\rangle\rangle$ has a strategy to enforce φ in the next state"; $\langle\langle\Delta\rangle\rangle G\varphi$ denotes "$\langle\langle\Delta\rangle\rangle$ has a strategy to enforce φ forever in the future"; and $\langle\langle\Delta\rangle\rangle\varphi_1 U \varphi_2$ represents "$\langle\langle\Delta\rangle\rangle$ has a strategy to enforce that φ_2 holds at some point in the future and φ_1 holds until then".

A variable appearing in a PATL formula is said to be *free* if it is not in the scope of a universal quantifier. A PATL formula is said to be a *sentence* if there are no free variables appearing in the formula. We here consider only PATL sentences. We say that a PATL sentence is an $\overline{m}$-indexed formula, where $\overline{m}$ is a k-tuple of natural numbers, if there are precisely $\overline{m}.i$ variables from VAR_i appearing in a cooperation modality or a propositional variable in the formula.

The key difference of PATL with respect to plain ATL is that while a formula in ATL expresses what a concrete group can achieve irrespective of the actions of all other agents outside the group, a PATL formula quantifies over the groups of concrete agents that can enforce the property independently of how many concrete agents are actually instantiated; i.e., in any system of any size.

Assume a CCGS $S(\overline{n}) = \langle \Sigma, Q, Q_0, A, \Pi, \pi, d, \delta \rangle$. We now describe the evaluation of PATL sentences on $S(\overline{n})$. This uses the notion of a *strategy*. A strategy for a concrete agent $i \in \Sigma_C$ is a function $f_i : \hat{Q}^+ \rightarrow \hat{A}$ such that for each finite sequence $\hat{\lambda} \in \hat{Q}^+$ we have that $f_i(\hat{\lambda}) \in \hat{d}(i, \hat{q})$, where $\hat{q}$ is the last state in $\hat{\lambda}$. Similarly, a strategy for a concrete agent $(i,j) \in \Sigma_T$ is a function $f_{(i,j)} : \hat{Q}^+ \rightarrow \hat{A}$ such that for each finite sequence $\hat{\lambda} \in \hat{Q}^+$ we have that $f_{(i,j)}(\hat{\lambda}) \in \hat{d}(\hat{i}, \hat{q})$, where $\hat{q}$ is the last state in $\hat{\lambda}$. A strategy is said to be *memoryless* if it only depends on the last state, i.e., $f_{ag}(\hat{q}_1, \ldots, \hat{q}_x, \hat{q}) = f_{ag}(\hat{q})$, for any $x \in \mathbb{N}$. In the following we consider memoryless strategies.

[2] A serial transition relation is induced by a given PCGS by assuming a *serial* action such that $serial \in \hat{d}(ag, q)$ and $\hat{\delta}(ag, q, serial, X) = q$, for each $q \in \hat{Q}$, $ag \in \hat{\Sigma}$, and $X \subseteq \hat{A}$.

Given a set Γ of concrete agents, a concrete state $q \in Q$, and an indexed set of strategies $F_\Gamma = \{f_{ag} : ag \in \Gamma\}$, the *outcomes* of F_Γ from q is defined as the set of infinite paths $out(q, F_\Gamma)$, where $q_0 q_1 \ldots \in out(q, F_\Gamma)$ iff $q_0 = q$, and for every $x \geq 0$ there is a joint action $\overline{a}$ such that: (i) $\overline{a}.ag = f_{ag}(q_0.ag \ldots q_x.ag)$ for all agents $ag \in \Gamma$; (ii) $\delta(q_x, \overline{a}) = q_{x+1}$.

We now define the satisfaction relation. We write $(S(\overline{n}), q) \models \varphi$ to mean that a formula φ is true at state q in $S(\overline{n})$. If $S(\overline{n})$ is clear, then we simplify the notation to $q \models \phi$.

Definition 3 (Satisfaction of PATL) *Let $S(\overline{n})$ be a CCGS, q a state in $S(\overline{n})$, (p, ag) a propositional variable, and $v \in VAR_i$ a variable symbol. The satisfaction relation $\models$ is inductively defined as follows:*

$q \models (p, ag)$	*iff*	$(p, ag) \in \Pi(q)$;
$q \models \neg\varphi$	*iff*	$q \not\models \varphi$;
$q \models \varphi_1 \wedge \varphi_1$	*iff*	$q \models \varphi_1$ *and* $q \models \varphi_2$;
$q \models \langle\langle\Gamma\rangle\rangle X\varphi$	*iff*	*for some* F_Γ *and all* $\lambda \in out(q, F_\Gamma)$ *we have that* $\lambda^1 \models \varphi$;
$q \models \langle\langle\Gamma\rangle\rangle\varphi_1 U \varphi_2$	*iff*	*for some* F_Γ *and all* $\lambda \in out(q, F_\Gamma)$, *there is* $i \geq 0$ *with* $\lambda^i \models \varphi_2$ *and* $\lambda^j \models \varphi_1$ *for all* $0 \leq j < i$;
$q \models \langle\langle\Gamma\rangle\rangle G\varphi$	*iff*	*for some* F_Γ *and all* $\lambda \in out(q, F_\Gamma)$ *and for all* $i \geq 0$ *we have that* $\lambda^i \models \varphi$;
$q \models \forall v\varphi$	*iff*	*for all* $ag \in \{(i,1), \ldots, (i, \overline{n}.i)\}$, $q \models \varphi[v \mapsto ag]$.

A PATL formula φ is said to be true in $S(\overline{n})$, denoted $S(\overline{n}) \models \varphi$, if $(S(\overline{n}), q) \models \varphi$ for every $q \in Q_0$.

PATL generalises indexed CTL [12], a parametric variant of CTL that introduces quantification operators over the system components. In addition to the next-time operator, the unrestricted nesting of the quantification operators can be used to represent the actual number of participants in the system [12], thereby making the parameterised model checking problem undecidable [11]. To circumvent this, indexed CTL typically excludes the next-time operator and is restricted to its prenex fragment in which all the quantifiers appear at the front of the formula [5]. In light of this, for the rest of the paper, we consider $\overline{m}$-indexed PATL formulae that comply to the following schema:

$$\forall v_{(1,1)} \ldots \forall v_{(k,\overline{m}.k)}$$
$$\left(\left(\bigwedge_{i,j} \neg(v_{(1,i)} = v_{(1,j)}) \wedge \ldots \wedge \bigwedge_{i,j} \neg(v_{(k,i)} = v_{(k,j)}) \right) \rightarrow \varphi \right)$$

where φ is a PATL formula with no quantifiers and without the next-time operator that is built from precisely the variables $v_{(i,1)}, \ldots, v_{(i,\overline{m}.i)}$, for each agent template $\hat{i}$. We simply write φ to denote an $\overline{m}$-indexed formula of the above schema.

The evaluation of an $\overline{m}$-indexed formula on a CCGS is determined by evaluating the conjunction of all its ground instantiations under any assignment for the variables. For example, assume a resource sharing protocol encoded as a PCGS with one agent template, and consider that want to check the property: "every agent has a strategy so that they eventually take the lock on the shared resource". We can express this property in PATL by the 1-indexed formula $\langle\langle\{v\}\rangle\rangle F(lock, v)$. When evaluated on a concrete system with two participants, the formula denotes the formula $\langle\langle\{(1,1)\}\rangle\rangle F(lock,(\mathit{1},\mathit{1})) \wedge \langle\langle\{(\mathit{1},\mathit{2})\}\rangle\rangle F(lock,(\mathit{1},\mathit{2}))$. Following the symmetric nature of these conjuncts, an $\overline{m}$-indexed formula can be evaluated by considering only its ground instantiation obtained by assigning pairwise distinct values to the variables in each VAR_i from the domain $\{(i,1),\ldots,(i,\overline{m}.i)\}$ [30]. For example, $\langle\langle\{v\}\rangle\rangle F(lock, v)$ can be evaluated by simply considering its ground instantiation $\langle\langle\{(1,1)\}\rangle\rangle F(lock,(\mathit{1},\mathit{1}))$. For the rest of the paper, an $\overline{m}$-indexed formula equivalently refers to its aforementioned ground instantiation.

3 Parameterised Model Checking

We now introduce a procedure to assess the correctness of a MAS with respect to a given PCGS independently of the number of agents in the system. This decision problem is generally known as the *parameterised model checking problem* [8].

Definition 4 (PMCP) *Given a PCGS $\hat{S}$ and an $\overline{m}$-indexed PATL formula φ, the parameterised model checking problem (PMCP) is the decision problem of determining whether the following holds:*

$$S(\overline{n}) \models \varphi \textit{ for every } \overline{n} \geq \overline{m}\,^{3}.$$

The PCMP is known to be undecidable for arbitrary systems [6]. However, restrictions can be imposed on the systems leading to decidable problems; these have been exploited in a variety of applications ranging from networking to MAS against temporal or epistemic specifications [13, 18, 30].

We prove below that the PMCP for PCGS is undecidable. To achieve this, we show that PCGS can simulate broadcast protocols [16], whose PMCP has been shown undecidable for liveliness properties [19]. This result shows the generality of systems that can be described by PCGS. Additionally, it will give us the intuition to define a special class of PCGS that enjoys a decidable PMCP.

A broadcast protocol is a tuple $B = (S, S_0, \Lambda, R)$, where S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $\Lambda = L \times \{!!\} \cup L \times \{??\}$ is the union of finite sets of output broadcast labels ($L \times \{!!\}$) and input broadcast labels ($L \times \{??\}$), and $R \subseteq S \times \Lambda \times S$ is a transition relation [16]. A concrete broadcast system $B(n)$ can be constructed from B and a parameter $n \in \mathbb{N}$ representing the number of processes in the system. The processes communicate via broadcast primitives, i.e., processes send messages that are received by all other processes. Formally, there is a transition from a global state g to a global state g' by means of label a if there is a process i such that $(g.i, a!!, g'.i) \in R$, and for all processes $j \neq i$, $(g.j, a??, g'.j) \in R$.

Theorem 1 *The PMCP for PCGS is undecidable.*

Proof sketch. Let $B = (S, S_0, \Lambda, R)$ be a broadcast protocol. We construct a PCGS $\hat{S} = \left\langle \hat{\Sigma}, \hat{Q}, \hat{q_0}, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \right\rangle$ that simulates B (see Figure 1):

- $\Sigma_C = \emptyset$, $\hat{\Sigma_T} = \{\hat{1}\}$; $\hat{Q} = S$; $\hat{q_0}(\hat{1}) = S_0$; $\hat{A} = \Lambda$; $\hat{\Pi}$ and $\hat{\pi}$ can be arbitrarily defined.
- $\hat{d}$ is defined as follows: for each $q \in S$ and $a \in \Lambda$ with $(q, a, q') \in R$, for some q', we have that $a \in \hat{d}(\hat{1}, q)$.

^{3} $\overline{n} \geq \overline{m}$ is a shortcut for $\overline{n}.i \geq \overline{m}.i$, for each $1 \leq i \leq k$.

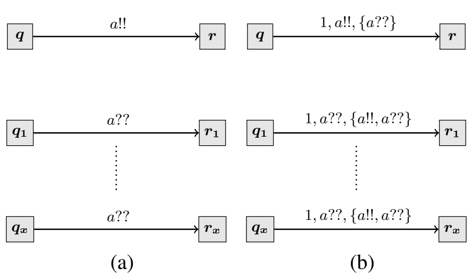


Figure 1: Simulation of broadcast protocols by PCGS.

- $\hat{\delta}$ is given by the following ^{4}: for each $(q, a!!, q') \in R$, we have that $\hat{\delta}(\hat{1}, q, a!!, \{a??\}) = q'$; and for each $(q, a??, q') \in R$, we have that $\hat{\delta}(\hat{1}, q, a??, \{a!!, a??\}) = q'$.

It follows that for every $n \in \mathbb{N}$, and for every sequence of states $\lambda \in (S^n)^+$, λ is a path in $B(n)$ iff λ is a path in $S(n)$. But the PMCP for broadcast protocols against liveliness properties is undecidable [16]. Since liveness can be expressed in PATL, it follows that the PMCP for PCGS against PATL specifications is also undecidable. ■

In the proof above note the crucial role of the transition $\hat{\delta}(\hat{1}, q, a!!, \{a??\}) = q'$. By the concrete semantics, the transition is enabled for a concrete agent if the agent exclusively performs $a!!$, and all other agents perform $a??$. Intuitively this encodes a situation where the agent takes the lock on the shared resource q'. We define the class of *mutual exclusion* PCGS on the basis of this interpretation.

We write $q \rightarrow_a q'$ to mean that there is a template transition $\hat{\delta}(\hat{ag}, q, a, X) = q'$ such that $a \in X$ for some agent template $\hat{ag}$. That is, if $q \rightarrow_a q'$, then an arbitrary number of agents from a certain template may perform the action a and move from state q to state q'. Assume $q \dashrightarrow_a q'$ to denote the latter, but with $a \notin X$. That is, if $q \dashrightarrow_a q'$, then precisely one agent following a certain template may perform the action a and move from state q to state q'. Call a template state q *initialisable* if for every template transition $\hat{\delta}(\hat{ag}, q, a, X) = q'$ we have that $q' \in \hat{q_0}(\hat{ag})$.

Definition 5 (Resource state) *A template state q is said to be a* resource state *if: (i) $\nexists(q', a).\, q' \rightarrow_a q$; (ii) $\exists!(q', a).\, q' \dashrightarrow_a q$; (iii) q is initialisable.*

Definition 6 (Non-resource state) *A template state q is said to be a* non-resource state *if $q' \rightarrow_a q$ by means of agent template $\hat{ag}$ implies $q' \dashrightarrow_a q$ by means of $\hat{ag}$, and $q' \dashrightarrow_a q$ by means of an agent template $\hat{ag}$ implies $q' \rightarrow_a q$.*

Intuitively, a template state is a resource state if: by conditions (i) and (ii), exactly one agent (in a concrete system) can take the lock on the shared resource represented by the state at any given time-step; by condition (iii) an agent always releases the lock on a shared resource. Differently, a non-resource state is a template state in which an arbitrary number of agents can move into at any given time step. Clearly, there may be template states that are neither resource states nor non-resource states.

Definition 7 (Mutual exclusion property) *A PCGS is said to satisfy the* mutual exclusion *property if the following conditions hold: (i) every state is either a resource state or a non-resource state; (ii) every initial template state is a non-resource state.*

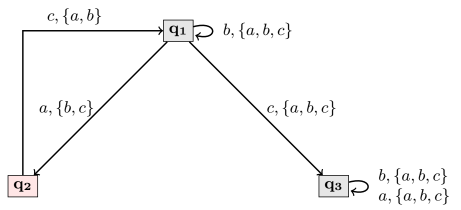


Figure 2: A PCGS violating the starvation freedom property. For clarity several transitions to non-resource states required by the mutual exclusion property are ommited in the figure.

A PCGS satisfying the mutual exclusion property may violate the so called *starvation freedom* property [21]. More specifically, the number of times an agent can access a shared resource may depend on the number of agents present in a concrete system. For example, consider the PCGS of one agent template $\hat{1}$ and zero concrete agents depicted in Figure 2. In a concrete system all agents are initially in state q_1. Assume that the concrete agent ag wants to access the shared resource represented by the resource state q_2. The agent has to perform the action a, and all other agents have to collectively perform the actions b and c. So, the agent ag goes to state q_2 and all the other agents move to states q_1 and q_3. Note that the agents in q_3 remain forever in q_3. To release the lock on the shared resource, the agent ag has to perform the action c and all the other agents have to collectively perform the actions a, b. Then, ag can take again the lock on the shared resource as described above. However, each time it does so it causes other agents to move from q_1 to q_3. Thus, eventually, no agents are left in q_1. At this point, ag cannot move from q_1 to q_2 since no agent can perform the action c. Hence, ag can access the shared resource only a finite number of times, the precise number depending on the number of agents in a concrete system.

The above behaviour is particularly problematic for parameterised model checking techniques. In counter abstraction it generates spurious paths that can be hard to identify [15, 28]. We thus restrict the analysis of PCGS to the class of PCGS that satisfy the mutual exclusion property and the starvation freedom property. The latter is defined as follows. Call a template transition $\hat{\delta}(\hat{i}, q, a, A) = q'$ *bad* on action b if $q \neq q'$, $b \in A$, and $\hat{\delta}(\hat{i}, q, a, A \setminus \{b\}) \neq q'$. In other words, a template transition from a state q to a different state is bad on an action b if a different action cannot be performed at the state without b being performed at the state. For example, the transition $\hat{\delta}(\hat{1}, q_1, a, \{b, c\}, q_2)$ of the PCGS shown in Figure 2 is bad on actions b and c. The starvation freedom property insists on these actions to be reflexive on the state they are performed, thereby "eliminating" the behaviour described above.

Definition 8 (Starvation freedom property) *A PCGS is said to satisfy the starvation freedom property if the following condition holds: if a transition $\hat{\delta}(\hat{i}, q, a, A) = q'$ is bad on an action b and $\hat{\delta}(\hat{j}, r, b, A) = r'$, for any $\hat{j}$, r, then $r = r'$.*

Definition 9 ($\mathbb{ME}$ class of PCGS) *The $\mathbb{ME}$ class of PCGS is the class of all PCGS that satisfy the mutual exclusion property and the starvation freedom property.*

Intuitively, the $\mathbb{ME}$ class describes resource allocation protocols in which every agent can eventually access a shared resource after requesting to do so. The importance of the $\mathbb{ME}$ class of systems is particularly significant and goes well beyond these protocols. Indeed, in the next section we show that the PMCP for $\mathbb{ME}$ MAS against PATL specifications is decidable.

4 Verifying the $\mathbb{ME}$ Class of PCGS

In this section we show the significance of the $\mathbb{ME}$ class of PCGS identified above. Specifically we give a procedure that solve. the PMCP for $\mathbb{ME}$ PCGS against PATL formulas. Given an $\mathbb{ME}$ PCGS and an $\overline{m}$-indexed formula, the method involves checking a particular abstract model $\mathcal{S}(\overline{m})$ against the formula. We show below that the abstract model $\mathcal{S}(\overline{m})$ finitely encodes all concrete instantiations of a given PCGS. It follows that the satisfaction of the formula on the abstract model implies the satisfaction of the formula on every concrete instantiation of the original PCGS. Conversely, if the specification is not satisfied on the abstract model $\mathcal{S}(\overline{m})$, there exists a concrete instantiation of the original PCGS that falsifies the formula.

Fix a PCGS $\hat{S} = \left\langle \hat{\Sigma}, \hat{Q}, \hat{q_0}, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \right\rangle \in \mathbb{ME}$ of z concrete agents $\Sigma_C = \{1, \ldots, z\}$ and k agent templates $\hat{\Sigma_T} = \left\{\hat{1}, \ldots, \hat{k}\right\}$. Let φ be an $\overline{m}$-indexed PATL formula. For $\overline{x} \in \mathbb{N}^k$, let $\Sigma_T(\overline{x})$ denote the set $\Sigma_T(\overline{x}) = \{(i, j) \colon 1 \leq i \leq k, 1 \leq j \leq \overline{x}.i\}$ of concrete agents.We now construct the abstract model $\mathcal{S}(\overline{m})$. An abstract state

$$\gamma = \big((q_{(1,1)}, \ldots, q_{(k,\overline{m}.k)}, q_1, \ldots, q_z), (X_1, \ldots, X_k)\big)$$

consists of a concrete component $\gamma.c = (q_{(1,1)}, \ldots, q_{(k,\overline{m}.k)}, q_1, \ldots, q_z)$ and an abstract component $\gamma.ab = (X_1, \ldots, X_k)$. The abstract state γ represents any concrete state g in an arbitrarily large system $S(\overline{n})$ in which: each concrete agent $(i, j) \in \Sigma_T(\overline{m})$ is at local state $q_{(i,j)}$; each concrete agent $i \in \Sigma_C$ is at local state q_i; for each agent template $\hat{i}$, $X_i = \{g.(i, j) \colon (i, j) \in \Sigma_T(\overline{n}) \setminus \Sigma_T(\overline{m})\}$ is the set of states for the agents not in $\Sigma_T(\overline{m})$. We write $\gamma.c.ag$ for the local state of the agent ag in γ. By $\gamma.ab.i$ we denote the abstract component associated with the agent template $\hat{i}$ in γ.

Definition 10 (Abstract model) *Given a PCGS $\hat{S}$ of k agent templates and z concrete agents, and an $\overline{m}$-indexed PATL specification φ, the abstract model $\mathcal{S}(\overline{m})$ is defined as the tuple $\mathcal{S} = \langle \mathcal{G}, \mathcal{I}, \mathcal{T}, \mathcal{V} \rangle$, where:*

- $\mathcal{G} \subseteq \hat{Q}^{|\Sigma_T(\overline{m})|} \times \hat{Q}^z \times (2^{\hat{Q}})^k$ *is the set of abstract states.*
- $\mathcal{I} = \left(\hat{q_0}(\hat{1})\right)^{\overline{m}.1} \times \left(\hat{q_0}(\hat{k})\right)^{\overline{m}.k} \times \hat{q_0}(1) \times \ldots \hat{q_0}(z) \times 2^{\hat{q_0}(\hat{1})} \times \ldots \times 2^{\hat{q_0}(\hat{k})}$ *is the set of initial abstract states.*
- $\mathcal{T} \subseteq \mathcal{G} \times \Lambda^k \times \mathcal{G}$ *is the abstract transition relation, where* $\Lambda = 2^{\hat{Q} \times \hat{A} \times 2^{\hat{A}} \times \hat{Q}}$ *is a set of* transition labels.
- $\mathcal{V} : \mathcal{G} \to 2^{(\hat{\Pi} \times \Sigma_C) \cup (\hat{\Pi} \times \Sigma_T(\overline{m}))}$ *is the labelling function defined by $(p, ag) \in \mathcal{V}(\gamma)$ iff $p \in \hat{\pi}(\gamma.c.ag)$.*

The concrete component $\gamma.c$ encodes the atomic propositions from which φ is built, whereas the abstract component $\gamma.ab$ encodes the ways an arbitrary number of agents may interfere with the state of $\gamma.c$. We make this precise with the definition of the abstract transition relation. Transitions from an abstract state represent transitions enabled from any concrete state represented by said abstract state. Each abstract transition is labelled by the template transitions taking place in a concrete representative transition. Specifically, for an abstract transition $(\gamma, (\Xi_1, \ldots, \Xi_k), \gamma')$, a label $(q, a, A, q') \in \Xi_i$ indicates that in a global transition at least one agent in $\Sigma_T(\overline{n}) \setminus \Sigma_T(\overline{m})$ is in state q; the agent performs the action a and moves to the state q', and all other agents perform the set of actions A.

[4] Note that the undecidability result in [19] is drawn under the assumption that for each $q \in S$, $a \in \Lambda$, there is exactly one state q' with $(q, a, q') \in R$.

Let $\Xi = \Xi_1 \cup \ldots \cup \Xi_k$. We write $Actions(\Xi) = \{a \colon \exists q, q', A.\, (q, a, A, q') \in \Xi\}$ for the set of actions performed by Ξ. We now define the abstract transition relation.

Definition 11 (Abstract transition relation) *The abstract transition relation $\mathcal{T}$ is defined as $(\gamma, (\Xi_1, \ldots, \Xi_k), \gamma') \in \mathcal{T}$ iff there is a joint action $\overline{a} \in \hat{A}^{|\Sigma_T(\overline{m})|+|\Sigma_C|}$ such that:*

- *for every concrete agent $ag = (i, j) \in \Sigma_T(\overline{m})$, we have $\overline{a}.ag \in \hat{d}(\hat{i}, \gamma.c.ag)$ and $\hat{\delta}(\hat{i}, \gamma.c.ag, \overline{a}.ag, Actions_{-ag}(\overline{a}) \cup Actions(\Xi)) = \gamma'.c.ag$.*
- *for every concrete agent $i \in \Sigma_C$, we have $\overline{a}.i \in \hat{d}(i, \gamma.c.i)$ and $\hat{\delta}(i, \gamma.c.i, \overline{a}.i, Actions_{-i}(\overline{a}) \cup Actions(\Xi)) = \gamma'.c.i$.*
- *for each agent template $\hat{i}$, for all $l = (q, a, A, q') \in \Xi_i$, we have $q \in \gamma.ab.i$, $q' \in \gamma'.ab.i$, $a \in \hat{d}(\hat{i}, q)$, $\hat{\delta}(\hat{i}, q, a, A) = q'$, and either $A = Actions(\overline{a})$ or $A = Actions(\overline{a}) \cup Actions(\Xi \setminus \{l\})$;*
- *for each agent template $\hat{i}$, for every $q \in \gamma.ab.i$, there is a transition label in Ξ_i to a state $q' \in \gamma'.ab.i$, and for every $q' \in \gamma'.ab.i$, there is a transition label in Ξ_i from a state $q \in \gamma.ab.i$.*

A path in $\mathcal{S}(\overline{m})$ is a sequence $\gamma_0\gamma_1, \ldots$ such that for every $i \geq 0$ there is a k-tuple of transition labels $(\Xi_1, \ldots, \Xi_k)$ with $(\gamma_i, (\Xi_1, \ldots, \Xi_k), \gamma_{i+1}) \in \mathcal{T}$.

The satisfaction of an $\overline{m}$-indexed formula φ on $\mathcal{S}(\overline{m})$ implies the satisfaction of φ on every concrete system $S(\overline{n})$ with $\overline{n} \geq \overline{m}$.

Lemma 1 *Let $\hat{S} \in \mathbb{ME}$ be a PCGS of k agent templates and z concrete agents. Let φ be an $\overline{m}$-indexed formula. Then, $\mathcal{S}(\overline{m}) \models \varphi$ implies $S(\overline{n}) \models \varphi$ for every $\overline{n}$ with $\overline{n} \geq \overline{m}$.*

Proof sketch. Let $\overline{n} \geq \overline{m}$. Define a mapping ζ from the set of concrete states in $S(\overline{n})$ to the set of abstract states in $\mathcal{S}(\overline{m})$:

$$\zeta(q) = (q.(1,1), \ldots, q.(k, \overline{m}.k), q.1, \ldots, q.z, \Xi_1, \ldots, \Xi_k)\,,$$

where $\Xi_1 = \{q.(1, j) \colon \overline{m}.1 + 1 \leq j \leq \overline{n}.1\}, \ldots, \Xi_k = \{q.(k, j) \colon \overline{m}.k + 1 \leq j \leq \overline{n}.k\}$. For each agent template $\hat{i}$, define a mapping ξ_i from concrete transitions to abstract transition labels as follows: for a concrete transition $\delta(q, \overline{a}) = q'$, $(r, a, A, r') \in \xi_i((q, \overline{a}, q'))$ iff there is a j with $\overline{m}.i + 1 \leq j \leq \overline{n}.i$ such that $q.(i, j) = r$, $\overline{a}.(i, j) = a$, $A = Actions_{-(i,j)}(\overline{a})$, and $q'.(i, j) = r'$.

We show that if $\zeta(q) = \gamma$ and $\gamma \models \varphi$, for an $\overline{m}$-indexed formula φ, then $q \models \varphi$. We do this by induction on φ.

Assume $(p, ag) \in \left(\hat{\Pi} \times \Sigma_C\right) \cup \left(\hat{\Pi} \times \Sigma_T(\overline{m})\right)$ and $\gamma \models (p, ag)$. Then, $(p, ag) \in \mathcal{V}(\gamma)$. Therefore, $(p, ag) \in \hat{\pi}(\gamma.c.ag)$, and therefore $(p, ag) \in \pi(q)$. Hence, $q \models p$.

The cases of $\varphi = \neg\varphi_1$, and $\varphi_1 \wedge \varphi_2$ are straightforward.

Suppose that $\varphi = \langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$ for a group of agents $\Gamma \subseteq \Sigma_C \cup \Sigma_T(\overline{m})$. Let $\gamma \models \varphi$. Consider a strategy $F_\Gamma = \{f_{ag} \colon ag \in \Gamma\}$ such that for all $\sigma \in out(\gamma, F_\Gamma)$, there is $i \geq 0$ with $\sigma^i \models \varphi_2$ and $\sigma^j \models \varphi_1$ for all $0 \leq j < i$. Choose the same strategy F_Γ for the group Γ of agents in $S(\overline{n})$. Let $\lambda = q_0 q_1 \ldots \in out(q, F_\Gamma)$, where $q_0 = q$. Assume a sequence $\overline{a}_0\overline{a}_1 \ldots$ of joint actions enabling the path λ; i.e, for each $i \geq 0$, $\delta(q_i, \overline{a_i}) = q_{i+1}$. It can be checked that for each $i \geq 0$, $(\delta(q_i), (\xi_1((q_i, \overline{a}, q_{i+1})), \ldots, \xi_k(q_i, \overline{a}, q_{i+1}))\,, q_{i+1}) \in \mathcal{T}$. Therefore, $\delta(\lambda) = \delta(q_0)\delta(q_1)\ldots$ is a path in $\mathcal{S}(\overline{m})$. Moreover, since $\lambda \in out(F_\Gamma, q)$, it follows that $\delta(\lambda) \in out(F_\Gamma, \gamma)$. Thus, there is $i \geq 0$ with $\delta(\lambda)^i \models \varphi_2$ and $\delta(\lambda)^j \models \varphi_1$ for all $0 \leq j < i$. By the inductive hypothesis, $\lambda^i \models \varphi_2$ and $\lambda^j \models \varphi_1$ for all $0 \leq j < i$. Hence, $q \models \phi$.

The case of $\varphi = \langle\langle\Gamma\rangle\rangle G\varphi_1$ can be shown similarly to the case of $\varphi = \langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$.

Therefore, we have shown that $\delta(q) = \gamma$ and $\gamma \models \varphi$ implies that $q \models \varphi$. As for every initial state q in $S(\overline{n})$, $\delta(q)$ is an initial state in $\mathcal{S}(\overline{m})$, it follows that $\mathcal{S}(\overline{m}) \models \varphi$ implies $S(\overline{n}) \models \varphi$. ■

Conversely, the falsification of an $\overline{m}$-indexed formula on $\mathcal{S}(\overline{m})$ implies the existence of a concrete system falsifying the formula.

Lemma 2 *Let $\hat{S} \in \mathbb{ME}$ be a PCGS of k agent templates and z concrete agents. Let φ be an $\overline{m}$-indexed formula. Then, $\mathcal{S}(\overline{m}) \not\models \varphi$ implies that there is $\overline{n} \geq \overline{m}$ with $S(\overline{n}) \not\models \varphi$.*

Proof sketch. Given an agent template $\hat{i}$, let $\hat{\delta}_i = \left\{(q, a, A, q) \colon \hat{\delta}(\hat{i}, q, a, A) = q\right\}$. For a tuple $t = (q, a, A, q)$, $t \in \hat{\delta}_i$, let λ_t be a finite concrete path such that there is an agent in state q in the last state of the path. Denote the concrete system in which λ_t occurs by $S(\overline{n_t})$; denote the agent that is in state q in λ_t by ag_t. Let $\overline{n} = \overline{m} + \sum_{t \in \hat{\delta}_1 \cup \ldots \cup \hat{\delta}_k} \overline{n_t}$, where $\overline{n_{t1}} + \overline{n_{t2}}$ denotes $(\overline{n_{t1}}.1 + \overline{n_{t2}}.1, \ldots, \overline{n_{t1}}.k + \overline{n_{t2}}.k)$. Given an agent template $\hat{i}$, let $\bigcup_t Ag_{i,t}$ be a partition of the set $\{(i, \overline{m}.i + 1), \ldots, (i, \overline{n}.i)\}$ of agents in $S(\overline{n})$ such that $|Ag_{i,t}| = \overline{n_t}.i$ for each $t \in \hat{\delta}_1 \cup \ldots \cup \hat{\delta}_k$. Assume a bijective mapping $\zeta_{i,t} : Ag_{i,t} \rightarrow \{(i, 1), \ldots, (i, \overline{n_t}.k)\}$ from the set $Ag_{i,t}$ of agents in $S(\overline{n})$ to the set $\{(i, 1), \ldots, (i, \overline{n_t}.k)\}$ of agents in $\mathcal{S}(\overline{n_t})$. Define a relation R between the states in $\mathcal{S}(\overline{m})$ and the states in $S(\overline{n})$ as follows: $(\gamma, q) \in R$ if: (i) $\gamma.c.(i, j) = q.(i, j)$ for each $(i, j) \in \Sigma_C \cup \Sigma_T(\overline{m})$; (ii) for each $\hat{i}$, t, every agent $ag \in Ag_{i,t}$ is in local state equal to the local state of the agent $\zeta_{i,t}(ag)$ in the last state of λ_t.

We show that if $(\gamma, q) \in R$ and $\gamma \not\models \varphi$, for an $\overline{m}$-indexed formula φ, then $q \not\models \varphi$. We do this by induction on φ. The atomic case and the cases of $\varphi = \neg\varphi_1$, and $\varphi_1 \wedge \varphi_2$ are straightforward. Suppose that $\varphi = \langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$ for a group of agents $\Gamma \subseteq \Sigma_C \cup \Sigma_T(\overline{m})$. Let $\gamma \not\models \varphi$. Then, for every strategy $F_\Gamma = \{f_{ag} \colon ag \in \Gamma\}$, there is $\sigma \in out(\gamma, F_\Gamma)$ that falsifies $\varphi_1 U \varphi_2$. Choose a strategy F_Γ and a path $\sigma = \gamma_0\gamma_1 \ldots \in out(\gamma_0, F_\Gamma)$, where $\gamma_0 = \gamma$, falsifying $\varphi_1 U \varphi_2$. Choose the same strategy F_Γ for the group Γ of agents in $S(\overline{n})$. We show that there is a path $\lambda = q_0 q_1 \ldots$, where $q_0 = q$, in $out(F_\Gamma, q)$ such that $(\gamma_i, q_i) \in R$, for every $i \geq 0$. For each $i \geq 0$, consider $(\gamma_i, (\Xi_1^i, \ldots, \Xi_k^i), \gamma_{i+1}) \in \mathcal{T}$ by means of the joint action $\overline{a_i}$ for the agents in $\Sigma_C \cup \Sigma_T(\overline{m})$. Then, λ results from the following joint actions at every time step i:

- Every agent ag in $\Sigma_C \cup \Sigma_T(\overline{m})$ performs the action $\overline{a_i}.ag$.
- Let $\Xi = \Xi_1^i \cup \ldots \cup \Xi_k^i$ and $A = Actions(\Xi) \cup Actions(\overline{a_i})$. Consider $Bad = \{a \colon \exists t \in \Xi.\, t \text{ is bad on } a\}$. Then, for each $a \in Bad$, choose a transition label $t = (q, a, A, q) \in \hat{\delta}_i$ (since $a \in Bad$, by the starvation freedom property, such a transition label exists for a state q and an agent template $\hat{i}$), and let the agent $\zeta_{i,t}^{-1}(ag_t)$ perform the action a.
- All the rest of the agents perform the *serial* action.

It can be checked the λ is as required. It follows that λ falsifies $\varphi_1 U \varphi_2$. Thus, $q \not\models \varphi$. The case of $\varphi = \langle\langle\Gamma\rangle\rangle G\varphi_1$ can be shown similarly to the case of $\varphi = \langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$.

Therefore, we have shown that $(\gamma, q) \in R$ and $\gamma \not\models \varphi$ implies $q \not\models \varphi$. Now for every initial state γ in $\mathcal{S}(\overline{m})$, there is an initial state q in $S(\overline{n})$ where: $\gamma.c.(i, j) = q.(i, j)$ for each $(i, j) \in \Sigma_C \cup \Sigma_T(\overline{m})$; for each $\hat{i}$, t, every agent $ag \in Ag_{i,t}$ is in local state equal to the local state of the agent $\zeta_{i,t}(ag)$ in the initial state of λ_t. From q there is a path to a state q' such that $(\gamma, q') \in R$. This path is defined as $\circ_{t \in \hat{\delta_1} \cup \ldots \cup \hat{\delta_k}} \lambda'_t$, where $\circ$ denotes string concatenation, and λ'_t is the path in which: for every agent template $\hat{i}$, each agent ag in $Ag_{i,t}$

performs the sequences of actions that the agent $\zeta_{i,t}(ag)$ performs in $S(\overline{n_t})$; every other agent performs the *serial* action. Consequently, as the agents in $\Sigma_C \cup \Sigma_T(\overline{m})$ stutter at their initial states in the path from q to q', $\mathcal{S}(\overline{m}) \not\models \varphi$ implies $S(\overline{n}) \not\models \varphi$. ■

Theorem 2 *Let $\hat{S} \in \mathbb{ME}$ be a PCGS and φ an $\overline{m}$-indexed formula. Then, the PMCP returns* true *iff $\mathcal{S}(\overline{m}) \models \varphi$.*

Proof ($\Rightarrow$) If $\mathcal{S}(\overline{m}) \not\models \varphi$, then, from Lemma 2, the PMCP returns *false*, which is a contradiction. ($\Leftarrow$) From Lemma 1.

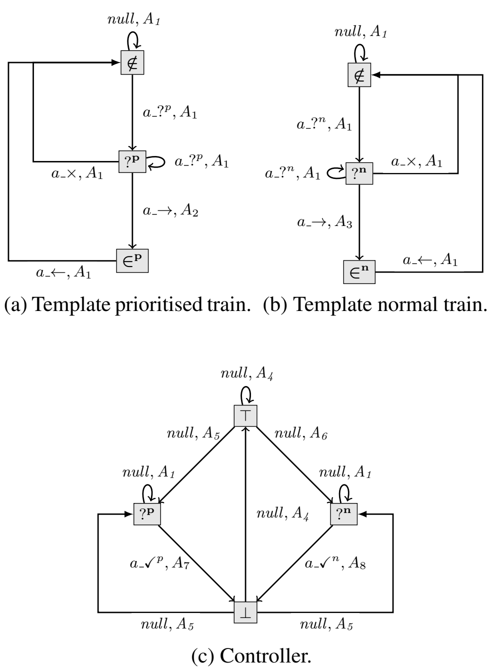
Theorem 2 is the main result of the paper. It states that, for a given $\overline{m}$-indexed PATL formula φ, the abstract model from Definition 10 is powerful enough to capture every possible instantiation of the PCGS that it is defined from. Since Definition 10 is entirely constructive, this gives us an immediate methodology for verifying any $\mathbb{ME}$ PCGS against any PATL specification: we simply construct the abstract model following Definition 10 and verify it against the specification φ. The result of this check is the result of the PMCP for the given PCGS against the PATL φ.

5 Prioritised Train-Gate-Controller

We exemplify the use of the methodology introduced in this paper on the prioritised train-gate controller (PTGC) [30], a parametric variant of the untimed version of the train-gate-controller [4, 20]. The system of PTGC is composed of a controller and an arbitrary number of two types of trains: *prioritised trains* and *normal trains*. Each train runs along a circular track and all tracks pass through a narrow tunnel. The tunnel can accommodate only one train to be in it at any time. Both sides of the tunnel are equipped with traffic lights, which can be either green or red. The controller operates the colour of the traffic lights to let the trains enter and exit the tunnel while complying with the following protocol: a normal train is given permission to enter the tunnel only when there is no pending request from a prioritised train to enter the tunnel.

We now model the PTGC as a PCGS $\mathfrak{G} = \left\langle \hat{\Sigma}, \hat{Q}, \hat{q_0}, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \right\rangle$, where (see Figure 3):

- $\Sigma_C = \{1\}$ and $\hat{\Sigma_T} = \{\hat{1}, \hat{2}\}$. Template agent $\hat{1}$ represents the prioritised trains, template agent $\hat{2}$ represents the normal trains, and concrete agent 1 represents the controller.
- $\hat{Q} = \{\not\in, ?^p, ?^n, \in^p, \in^n, \perp, \top\}$, where:
  - $\not\in$ represents that a train is not in the tunnel and it has not requested to enter the tunnel.
  - $?^p$ ($?^n$, respectively) represents that a prioritised (normal, respectively) train has requested to enter the tunnel.
  - $\in^p$ ($\in^n$, respectively) represents that a prioritised (normal, respectively) train is in the tunnel.
  - $\perp$ represents the traffic lights having colour red.
  - $\top$ represents the traffic lights having colour green.
- $\hat{q_0} = \{\hat{1} \mapsto \{\not\in\}, \hat{2} \mapsto \{\not\in\}, 1 \mapsto \{\top\}\}$.
- $\hat{A} = \{null, a\_?^p, a\_?^n, a\_\checkmark^p, a\_\checkmark^n, a\_\times, a\_\rightarrow, a\_\leftarrow\}$, where:
  - $a\_?^p$ ($a\_?^n$, respectively) represents a prioritised (normal, respectively) train making a request to enter the tunnel.
  - $a\_\times$ represents a train relinquishing its request to enter the tunnel.



(a) Template prioritised train. (b) Template normal train.

(c) Controller.

Figure 3: The PCGS of the PTGC.

  - $a\_\checkmark^p$ ($a\_\checkmark^n$, respectively) represents the controller accepting a prioritised (normal, respectively) train's request to enter the tunnel.
  - $a\_\rightarrow$ ($a\_\leftarrow$, respectively) represents a train entering (exiting, respectively) the tunnel.
- $\hat{\Pi} = \{tp, np\}$, $\hat{\pi}(\in^p) = \{tp\}$, and $\hat{\pi}(\in^n) = \{tn\}$. A prioritised (normal, respectively) train has entered the tunnel.
- - $\hat{d}(\hat{1}, \not\in) = \{null, a\_?^p\}$. Whenever a prioritised train is not in the tunnel, it can choose to do nothing or it can make a request to enter the tunnel.
  - $\hat{d}(\hat{1}, ?^p) = \{a\_?^p, a\_\times, a\_\rightarrow\}$. The train has already made a request to enter the tunnel but the request has not yet been granted. The train can relinquish its request, or make a new request, or, if the controller grants access to the tunnel, enter the tunnel.
  - $\hat{d}(\hat{1}, \in^p) = \{a\_\leftarrow\}$. Whenever a train is in the tunnel, it can only exit the tunnel.
  - $\hat{d}$ is similarly defined for normal trains.
  - $\hat{d}(1, \top) = \hat{d}(1, \perp) = \{null\}$.
  - $\hat{d}(1, ?^p) = \{null, a\_\checkmark^p\}$, $\hat{d}(1, ?^n) = \{null, a\_\checkmark^n\}$. The controller can either delay a request from a train to enter the tunnel, or it can accept it.
- - $\hat{\delta}(\hat{1}, \not\in, null, A_1) = \not\in$ and $\hat{\delta}(\hat{1}, \not\in, a\_?^p, A_1) = ?^p$, where $A_1 \subseteq \hat{A}$. A prioritised train may choose to remain out of the tunnel or it may choose to request to enter the tunnel irrespectively of the other agents' actions. Similarly, $\hat{\delta}(\hat{1}, ?^p, a\_?^p, A_1) = ?^p$, $\hat{\delta}(\hat{1}, ?^p, a\_\times, A_1) = \not\in$, and $\hat{\delta}(\hat{1}, \in^p, a\_\leftarrow, A_1) = \not\in$. These transitions are similarly defined for normal trains.
  - $\hat{\delta}(\hat{1}, ?^p, a\_\rightarrow, A_2) = \in^p$, where $\{a\_\checkmark^p\} \subseteq A_2 \subseteq \hat{A} \backslash \{a\_\checkmark^n\}$. A prioritised train can enter the tunnel if the controller accepts a prioritised request and no other train is entering the tunnel

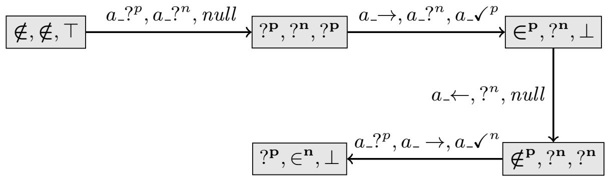

$\notin, \notin, \top$
$a\_?^p, a\_?^n, null$
$?^{\mathbf{p}}, ?^{\mathbf{n}}, ?^{\mathbf{p}}$
$a\_\!\rightarrow, a\_?^n, a\_\checkmark^p$
$\in^{\mathbf{p}}, ?^{\mathbf{n}}, \bot$
$a\_\!\leftarrow, ?^n, null$
$a\_?^p, a\_\rightarrow, a\_\checkmark^n$
$\notin^{\mathbf{p}}, ?^{\mathbf{n}}, ?^{\mathbf{n}}$
$?^{\mathbf{p}}, \in^{\mathbf{n}}, \bot$

Figure 4: Fragment of the CCGS $\mathfrak{G}((1,1))$.

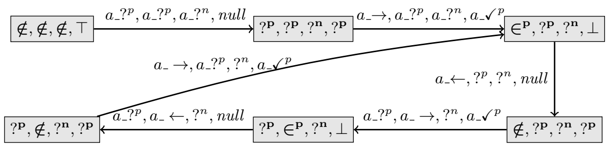

$\notin, \notin, \notin, \top$
$a\_?^p, a\_?^p, a\_?^n, null$
$?^{\mathbf{p}}, ?^{\mathbf{p}}, ?^{\mathbf{n}}, ?^{\mathbf{p}}$
$a\_\!\rightarrow, a\_?^p, a\_?^n, a\_\checkmark^p$
$\in^{\mathbf{p}}, ?^{\mathbf{p}}, ?^{\mathbf{n}}, \bot$
$a\_\rightarrow, a\_?^p, ?^n, a\_\checkmark^p$
$a\_\!\leftarrow, ?^p, ?^n, null$
$?^{\mathbf{p}}, \notin, ?^{\mathbf{n}}, ?^{\mathbf{p}}$
$a\_?^p, a\_\leftarrow, ?^n, null$
$?^{\mathbf{p}}, \in^{\mathbf{p}}, ?^{\mathbf{n}}, \bot$
$a\_?^p, a\_\rightarrow, ?^n, a\_\checkmark^p$
$\notin, ?^{\mathbf{p}}, ?^{\mathbf{n}}, ?^{\mathbf{p}}$

Figure 5: Fragment of the CCGS $\mathfrak{G}((2,1))$.

at the same time. Similarly, $\hat{\delta}(\hat{2}, ?^n, a\_\!\rightarrow, A_3) = \in^n$, where $\{a\_\checkmark^n\} \subseteq A_3 \subseteq \hat{A} \setminus \{a\_\checkmark^p\}$.

- $\hat{\delta}(1, \top, null, A_4) = \top$, where $A_4 \subseteq \hat{A} \setminus \{a\_?^p, a\_?^n\}$.
- $\hat{\delta}(1, \top, null, A_5) = ?^p$, where $\{a\_?^p\} \subseteq A_5 \subseteq \hat{A}$. If both prioritised and normal trains have requested to enter the tunnel, then the controller can only accept prioritised requests. Otherwise, if only normal requests have been made, then the controller can accept normal requests: $\hat{\delta}(1, \top, null, A_6) = ?^n$, where $\{a\_?^n\} \subseteq A_6 \subseteq \hat{A} \setminus \{a\_?^p\}$.
- $\hat{\delta}(1, ?^p, null, A_1) = ?^p$, $\hat{\delta}(1, ?^n, null, A_1) = ?^n$). The controller can delay handling a request irrespectively of the other agents' actions.
- $\hat{\delta}(1, ?^p, a\_\checkmark^p, A_7) = \bot$ and $\hat{\delta}(1, ?^n, a\_\checkmark^n, A_7) = \bot$, where $\{a\_\!\rightarrow\} \subseteq A_7 \subseteq \hat{A}$.
- $\hat{\delta}(1, \bot, null, A_4) = \top$, $\hat{\delta}(1, \bot, null, A_5) = ?^p$, and $\hat{\delta}(1, \bot, null, A_6) = ?^n$.

Any concrete system is generated by considering copies of the templates above. For example, a fragment of the concrete system of one prioritised train and one normal train is depicted in Figure 4; a fragment of the concrete system of two prioritised trains and one normal train is shown in Figure 5. In the figures each concrete state represents, from left to right, the local states of the prioritised trains, the local state of the normal train, and the local state of the controller. The tuples of actions are similarly read.

We now exemplify the use of PATL to reason about MAS with arbitrary many agents. We refer to [3, 4, 20] for several specifications of interest in plain ATL concerning the train-gate-controller. Differently from these works, where the overall system is composed of two trains, we can here represent specifications concerning the strategic properties of an unbounded number of trains. In particular, we are interested in formulating whether exactly one prioritised train can ensure that no normal trains can enter the tunnel irrespective of how many other trains (normal or prioritised) compose the system. This is expressed by the PATL formula

$$\varphi_{PTGC1} = \forall v\_1 \forall u\_2 \langle\!\langle \{v\} \rangle\!\rangle G \neg(tn, u),$$

where v_i indicates that the variable v is in VAR_i. Further, we would like to check whether at least two prioritised trains have a joint strategy to enforce the above property. This is expressed by the PATL formula

$$\varphi_{PTGC2} = \forall v\_1 \forall u\_1 \forall z\_2 \langle\!\langle \{v, u\} \rangle\!\rangle G \neg(tn, z).$$

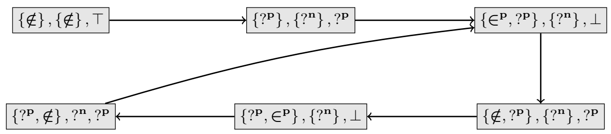

$\{\notin\}, \{\notin\}, \top$
$\{?^{\mathbf{p}}\}, \{?^{\mathbf{n}}\}, ?^{\mathbf{p}}$
$\{\in^{\mathbf{p}}, ?^{\mathbf{p}}\}, \{?^{\mathbf{n}}\}, \bot$
$\{?^{\mathbf{p}}, \notin\}, ?^{\mathbf{n}}, ?^{\mathbf{p}}$
$\{?^{\mathbf{p}}, \in^{\mathbf{p}}\}, \{?^{\mathbf{n}}\}, \bot$
$\{\notin, ?^{\mathbf{p}}\}, \{?^{\mathbf{n}}\}, ?^{\mathbf{p}}$

Figure 6: Fragment of the abstract model for the PTGC.

Finally, we would like to assess whether the controller has a strategy to ensure that precisely one train can be in the tunnel at any time, as expressed by the following PATL formula:

$$\begin{aligned}\varphi_{PTGC3} = &\forall v\_1 \forall u\_1 \forall z\_2 \forall w\_2 G \langle\!\langle \{1\} \rangle\!\rangle \\ &((tp, v) \rightarrow (\neg(tp, u) \wedge \neg(tn, z)) \wedge \\ &(tn, z) \rightarrow (\neg(tp, u) \wedge \neg(tn, w)))\end{aligned}$$

It is easy to check that $\mathfrak{G}$ satisfies the mutual exclusion property and the starvation freedom property. Therefore, $\mathfrak{G} \in \mathbb{ME}$, and therefore we can assess the correctness of the PTGC against the formulae φ_{PTGCi}, $1 \leq i \leq 3$, as per Theorem 2. A fragment of the abstract model that represents the fragment of the concrete system $\mathfrak{G}((2,1))$ (Figure 5) is shown in Figure 6. A state in the figure depicts, from left to right, the abstract component of the prioritised train, the abstract component of the normal train, and the local state of the controller. For brevity, the concrete components, of both the trains, and the transition labels are ommited in the figure.

The abstract model and the specifications can be checked by an ATL model checker; this would return *false* for φ_{PTGC1}, and *true* for φ_{PTGC2} and φ_{PTGC3}. Indeed, as it can be observed in Figures 5 and 6, any pair of prioritised trains can ensure that no normal trains can enter the tunnel: they can simply request access to the tunnel; since they are prioritised over normal trains, one of them will enter the tunnel while the other continues on requesting access to the tunnel, thus preventing a normal train to enter the tunnel. A single train, however, does not have a strategy to ensure this property, as it can be observed by the counterexample shown in Figure 4.

6 Conclusions

In this paper we put forward a technique for the parameterised verification of MAS against specifications expressing strategic behaviour of the agents. To achieve this we introduced PCGS, a novel parameterised extension of CGS (the usual semantics of ATL); PATL, a parameterised version of ATL; and defined the parameterised model checking problem for MAS given in PCGS against specifications expressed in PATL. The proof of undecidability of the PMCP for PCGS against PATL specifications enabled us to identify an expressive class of PCGS for which a decidable counter abstraction methodology could be given. This enabled us to verify unbounded MAS against strategic specifications, as the example of the prioritised train-gate-controller demonstrated.

In future work we would like to extend the specifications supported by the technique here introduced. For example [26, 27] address parameterised verification against epistemic specifications. It would be of interest to develop a technique that can account for both epistemic and strategic specifications such as those discussed here. However, the counter abstraction technique presented in [27] cannot seemingly be extended to ATL modalities. Moreover, the one here devised for ATL cannot be applied to epistemic modalities. More work is required to solve this problem.

Acknowledgments. This research was funded by the EPSRC under grant EP/I00520X/1 and a Doctoral Prize Fellowship.

REFERENCES

[1] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge, 'Knowledge and ability', in *Handbook of Logics for Knowledge and Belief*, College Publications, (2015).

[2] R. Alur, L. de Alfaro, R. Grosu, T. Henzinger, A. Thomas, M. Kang, C. Kirsch, R. Majumdar F. Mang, and B-Y. Wang, 'jMocha: A model checking tool that exploits design structure', in *Proceedings of the 23rd International Conference on Software Engineering (ICSE01)*, pp. 835–836. IEEE, (2001).

[3] R. Alur, L. de Alfaro, T. Henzinger, S. Krishnan, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran, 'MOCHA user manual', Technical report, University of California at Berkeley, (2000).

[4] R. Alur, T. A. Henzinger, and O. Kupferman, 'Alternating-time temporal logic', *Journal of the ACM*, **49**(5), 672–713, (2002).

[5] B. Aminof, S. Jacobs, A. Khalimov, and S. Rubin, 'Parameterized model checking of token-passing systems', in *Proceedings of the 15th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI14)*, volume 8318 of *Lecture Notes in Computer Science*, pp. 262–281. Springer, (2014).

[6] K.R. Apt and D. C. Kozen, 'Limits for automatic verification of finite-state concurrent systems', *Information Processing Letters*, **22**(6), 307–309, (1986).

[7] F. Belardinelli, D. Grossi, and A. Lomuscio, 'Finite abstractions for the verification of epistemic properties in open multi-agent systems', in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI15)*, pp. 854–860. AAAI Press, (2015).

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

[9] N. Bulling and W. Jamroga, 'Comparing variants of strategic ability: how uncertainty and memory influence general properties of games', *Autonomous Agents and Multi-Agent Systems*, **28**(3), 474–518, (2014).

[10] K. Chatterjee, T. Henzinger, and N. Piterman, 'Strategy logic', in *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR07)*, volume 4703, pp. 59–73, (2007).

[11] E. Clarke, M. Talupur, T. Touili, and H. Veith, 'Verification by network decomposition', in *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR04)*, volume 3170 of *Lecture Notes in Computer Science*, 276–291, Springer, (2004).

[12] E.M. Clarke, O. Grumberg, and M.C. Browne, 'Reasoning about networks with many identical finite state processes', *Information and Computation*, **81**(1), 13–31, (1989).

[13] E.M. Clarke, M. Talupur, and H. Veith, 'Proving ptolemy right: The environment abstraction framework for model checking concurrent systems', in *Proceedings of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS08)*, volume 4963 of *Lecture Notes in Computer Science*, pp. 33–47. Springer, (2008).

[14] C. Dima and F. Tiplea, 'Model-checking ATL under imperfect information and perfect recall semantics is undecidable', *CoRR*, **abs/1102.4225**, (2011).

[15] E. A. Emerson and K. S. Namjoshi, 'Automatic verification of parameterized synchronous systems', in *Proceedings of the 8th International Conference one Computer Aided Verification (CAV96)*, volume 1102 of *Lecture Notes in Computer Science*, pp. 87–98. Springer, (1996).

[16] E. A. Emerson and K. S. Namjoshi, 'On model checking for non-deterministic infinite-state systems', in *Proceedings of 13th International Symposium on Logic in Computer Science (LICS98)*, pp. 70–80. IEEE, (1998).

[17] E.A. Emerson and V. Kahlon, 'Model checking guarded protocols', in *Proceedings of the 14th International Symposium on Logic in Computer Science (LICS03)*, pp. 361–370. IEEE, (2003).

[18] E.A. Emerson and K.S. Namjoshi, 'Reasoning about rings', in *Proceedings of the 22nd Annual Sigact-Aigplan on Principles of Programming Languages (POPL95)*, pp. 85–94. Pearson Education, (1995).

[19] J. Esparza, A. Finkel, and R. Mayr, 'On the verification of broadcast protocols', in *Proceedings of the 10th International Symposium on Logic in Computer Science (LICS99)*, pp. 352–359. IEEE, (1999).

[20] W. van der Hoek and M. Wooldridge, 'Tractable multiagent planning for epistemic goals', in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS02)*, pp. 1167–1174. ACM Press, (2002).

[21] M. R. A. Huth and M. D. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems (2nd edition)*, Cambridge University Press, Cambridge, England, 2004.

[22] W. Jamroga and J. Dix, 'Model checking abilities under incomplete information is indeed δ_p^2-complete', in *Proceedings of the 4th European Workshop on Multi-Agent Systems EUMAS'06*, pp. 14–15. Citeseer, (2006).

[23] G. Jonker, *Feasible Strategies in Alternating-time Temporal Epistemic Logic*, Master's thesis, University of Utrech, The Netherlands, 2003.

[24] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny, 'Verics 2007 - a model checker for knowledge and real-time', *Fundamenta Informaticae*, **85**(1), 313–328, (2008).

[25] P. Kouvaros and A. Lomuscio, 'Automatic verification of parametrised interleaved multi-agent systems', in *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS13)*, pp. 861–868. IFAAMAS, (2013).

[26] P. Kouvaros and A. Lomuscio, 'A cutoff technique for the verification of parameterised interpreted systems with parameterised environments', in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI13)*, pp. 2013–2019. AAAI Press, (2013).

[27] P. Kouvaros and A. Lomuscio, 'A counter abstraction technique for the verification of robot swarms', in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15)*, pp. 2081–2088. AAAI Press, (2015).

[28] P. Kouvaros and A. Lomuscio, 'Verifying emergent properties of swarms', in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI15)*, pp. 1083–1089. AAAI Press, (2015).

[29] P. Kouvaros and A. Lomuscio, 'Formal verification of opinion formation in swarms', in *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS16)*, pp. 1200–1209. IFAAMAS, (2016).

[30] P. Kouvaros and A. Lomuscio, 'Parameterised verification for multi-agent systems', *Artificial Intelligence*, **234**, 152–189, (2016).

[31] A. Lomuscio, H. Qu, and F. Raimondi, 'MCMAS: A model checker for the verification of multi-agent systems', *Software Tools for Technology Transfer*, (2015). http://dx.doi.org/10.1007/s10009-015-0378-x.

[32] F. Mogavero, A. Murano, and L. Sauro, 'On the boundary of behavioral strategies', in *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS2013)*, pp. 263–272. IEEE, (2013).

[33] Marco Montali, Diego Calvanese, and Giuseppe De Giacomo, 'Verification of data-aware commitment-based multiagent system', in *Proceedings of the 14th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS14)*, pp. 157–164. IFAAMAS, (2014).

[34] C. Nardini, B. Kozma, and A. Barrat, 'Whos talking first? consensus or lack thereof in coevolving opinion formation models', *Physical review letters*, **100**(15), 158701, (2008).

[35] A. S. Rao and M. P. Georgeff, 'Decision procedures for BDI logics', *Journal of Logic and Computation*, **8**(3), 293–343, (1998).

[36] E. Semsar-Kazerooni and K. Khorasani, 'Multi-agent team cooperation: A game theory approach', *Automatica*, **45**(10), 2205–2213, (2009).