Parameterised Model Checking for Alternating-time Temporal Logic

Panagiotis Kouvaros\textsuperscript{1} and Alessio Lomuscio\textsuperscript{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 irrespective 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 specifications in PATL. We conclude in Section 6, where we discuss possible future work.

\textsuperscript{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., if 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 \(\Delta^{3}_T\) [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 \(\hat{S} = \langle \hat{\Sigma}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\sigma}, \hat{d}, \hat{\delta} \rangle\), where:

- \(\hat{\Sigma} = \Sigma_T \cup \Sigma_C\), where \(\Sigma_T = \{1, \ldots, \hat{k}\}\) is a finite, nonempty set of agent templates and \(\Sigma_C = \{1, \ldots, \hat{z}\}\) is a finite set of concrete agents.
- \(\hat{Q}\) is a nonempty, and finite set of local states for the agents.
- \(\hat{q}_0 = \Sigma \rightarrow 2^{\hat{Q}}\) describes a set of initial local states for each of the agents.
- \(\hat{\Pi} = \hat{\Pi} = \hat{\Pi}\) is a nonempty, and finite set of actions available to the agents.
- \(\hat{d} : \hat{\Sigma} \times \hat{Q} \rightarrow 2^{\hat{A}}\) is a labelling function on the states.
- \(\hat{\delta} : \hat{\Sigma} \times \hat{Q} \times \hat{A} \rightarrow \hat{\Sigma} \times \hat{Q} \times \hat{A}\) 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 \(\pi \in \mathbb{N}^k\) be a vector of parameters for the system. Consider \(\pi_i\) to be the \(i\)-th component in \(\pi\). We now define the \(\pi\)-st concrete instantiation \(S(\pi)\) of \(\hat{S}\). \(S(\pi)\) is a concurrent game structure resulting from the composition of the concrete agents \(\Sigma_C\) with \(\pi_i\) instantiations \(\{(i, 1), \ldots, (i, \pi_i)\}\) of each agent template \(i\). Let \(\Sigma\) denote the set of all concrete agents. The global states of \(S(\pi)\) 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 \(\pi\) and an agent \(ag \in \Sigma\), we write \(\pi.ag\) for the action of \(ag\) in \(\pi\). By \(Actions(\pi)\), we denote the set \(\pi.ag: ag \in \Sigma\) of actions for all the agents in \(\pi\), by \(Actions_{-ag}(\pi)\), we denote the set \(\pi.ag: ag \in \Sigma\) of actions for all the agents other than \(ag\) in \(\pi\).

**Definition 2 (Concrete concurrent game structure)** Let \(\hat{S} = \langle \hat{\Sigma}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\sigma}, \hat{d}, \hat{\delta} \rangle\) be a PCGS of \(k\) agent templates and \(\pi\) concrete agents. Let \(\pi \in \mathbb{N}^k\). A concrete concurrent game structure (CCGS) is a tuple \(S(\pi) = (\Sigma, Q, \bar{Q}, \bar{A}, \bar{\Pi}, \bar{\sigma}, \bar{d}, \bar{\delta})\), where:

- \(\Sigma = \Sigma_T \cup \Sigma_C\), where \(\Sigma_T = \{(i, j) : 1 \leq i \leq k, 1 \leq j \leq \pi_i\}\) is the set of concrete agents instantiated from the templates.
- \(Q = Q^{\Sigma_C}\) is the set of concrete states.
- \(\bar{Q}_0 = (q_0(1))^{\pi_1} \times \cdots \times (q_0(\hat{k}))^{\pi_{\hat{k}}} \times (q_0(1)) \times \cdots (q_0(z))\) is the set of initial concrete states.
- \(\bar{A} = \bar{A}^{\Sigma_C}\) is the set of joint actions.
- \(\bar{\Pi} = \bar{\Pi} \times \Sigma\).
- \(\pi : Q \rightarrow \bar{Q}\) is defined as \((p, ag) \in \pi(q)\) iff \(p \in \pi(\hat{q}.ag)\).
- \(\bar{d} : Q \rightarrow \bar{A}\) gives the set of joint actions available at a state of the system: \(\pi \in \bar{d}(q)\)

\[\bar{d}(i, q.i) \text{ for every agent } i \in \Sigma_C, \text{ and } \pi.(i, j) \in \bar{d}(i, q.(i, j)) \text{ for every agent } (i, j) \in \Sigma_T.\]

\[\bar{\delta} : Q \times \bar{A} \rightarrow Q \text{ is the temporal transition function of the system: }\]

\[\bar{\delta}(q, \bar{a}) = q'.\pi \in \bar{d}(q), \text{ for every agent } i \in \Sigma_C, \text{ and } \]

\[\bar{\delta}(q.(i, j), \bar{\pi}(i, j), Actions_{-\pi}(\bar{\pi})) = q'.(i, j) \text{ for every agent } (i, j) \in \Sigma_T.\]

So a PCGS generates different CCGSs 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 \(\lambda = q_0q_1 \cdots q_n\) such that for every \(i \geq 0\) there is a joint action \(\pi\) with \(\delta(q_i, \pi) = q_{i+1}\). For a path \(\lambda\) and
i \geq 0$, we write $\lambda^i$ to denote the suffix $q_iq_{i+1}\ldots$ of $\lambda$. 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 $\langle\Gamma\rangle \varphi$, where $\Gamma$ is a group of agents, expressing that $\Gamma$ has a cooperative strategy to enforce an LTL formula $\varphi$ irrespective of how the other agents act.

Let $\mathcal{VAR} = \mathcal{VAR}_1 \cup \ldots \cup \mathcal{VAR}_k$ be the union of disjoint sets of variable symbols, where each $\mathcal{VAR}_i$ is associated with agent template $i$. To reason about unbound groups of agents, we here define parameterised alternating-time temporal logic (PATL), a variation of ATL, where: (i) $\Gamma$ is not a subset of agents from a predetermined set as in ATL, but it is a subset $\Delta \subseteq \Sigma_C \cup \mathcal{VAR}$ of the union of the set of concrete agents $\Sigma_C$ and the set of variables $\mathcal{VAR}$; (ii) the atomic propositions are indexed by the variables in $\mathcal{VAR}$. The domain of a variable $v \in \mathcal{VAR}_i$ appearing in a formula $\varphi$ depends on the CCGS on which $\varphi$ is evaluated; if $\varphi$ is evaluated on $S(\pi)$, then the potential set of values for $v$ is $\{(i, 1), \ldots, (i, \pi, i)\}$. Intuitively, PATL formulas quantify over the concrete agents. For instance, $\forall v\langle\{v\}\rangle \varphi$, where $v \in \mathcal{VAR}_i$, expresses that every concrete agent from template $i$ can enforce $\varphi$ independently of the action performed by however many other agents are instantiated from each template.

Given a set $\Sigma$ of concrete agents, a set $\mathcal{VAR} = \mathcal{VAR}_1 \cup \ldots \cup \mathcal{VAR}_k$ of variable symbols, and a set $\Pi$ of propositional variables, PATL formulae are defined by the following BNF grammar:

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

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

We abbreviate $\langle\Delta\rangle(U\varphi)$ as $\langle\Delta\rangle F\varphi$. The formula $\langle\Delta\rangle X\varphi$ is read as “$\varphi$ has a strategy to enforce $\varphi$ in the next state”; $\langle\Delta\rangle G\varphi$ denotes “$\varphi$ has a strategy to enforce $\varphi$ forever in the future”; and $\langle\Delta\rangle(U\varphi)(U\varphi)$ represents “$\varphi$ has a strategy to enforce that $\varphi$ holds at some point in the future and $\varphi$ 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 $\pi, i$-indexed formula, where $\pi, i$ is a $k$-tuple of natural numbers, if there are precisely $\pi, i$ variables from $\mathcal{VAR}_i$ 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 $S(\pi) = (\Sigma, C, Q_0, A, \Pi, \pi, d, \delta)$. We now describe the evaluation of PATL sentences on $S(\pi)$. This uses the notion of a strategy. A strategy for a concrete agent $i \in \Sigma_C$ is a function $f_i : Q_i^+ \to A$ such that for each finite sequence $\hat{\lambda} \in Q_i^+$ we have that $f_i(\hat{\lambda}) \in d(i, \hat{\lambda})$, where $\hat{\lambda}$ is the last state in $\lambda$. Similarly, a strategy for

A serial transition relation is induced by a given PCGS by assuming a serial action such that $serial \in d(ag, q)$ and $d(ag, q, serial, X) = q$, for each $q \in Q, ag \in \Sigma, X \subseteq A$. A PATL formula $\varphi$ is said to be true in $S(\pi)$, denoted $S(\pi) \models \varphi$, if $(S(\pi), q) \models \varphi$ 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 prefix 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 $\pi, i$-indexed PATL formulae that comply to the following schema:

$$\forall v_{(1,1)} \ldots \forall v_{(k,\pi, k)} \left(\bigwedge_{i,j} \neg v_{(i,j)} \land \bigwedge_{i,j} \right) \varphi$$
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 \( \{\{v\}\} F(\text{lock}, v) \). When evaluated on a concrete system with two participants, the formula denotes the formula \( \{\{(1, 1)\}\} F(\text{lock}, (1, 1)) \land \{\{(1, 2)\}\} F(\text{lock}, (1, 2)) \). Following the symmetric nature of these conjuncts, an \( \pi \)-indexed formula can be evaluated by considering only its ground instantiation obtained by assigning pairwise distinct values to the variables in each \( \forall \mathcal{R}_i \) from the domain \( \{(i, 1), \ldots, (i, \pi, i)\} \) [30]. For example, \( \{\{v\}\} F(\text{lock}, v) \) can be evaluated by simply considering its ground instantiation \( \{\{(1, 1)\}\} F(\text{lock}, (1, 1)) \). For the rest of the paper, an \( \pi \)-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 \( \pi \)-indexed PATL formula \( \varphi \), the parameterised model checking problem (PMCP) is the decision problem of determining whether the following holds:

\[
S(\pi) \models \varphi \text{ for every } \pi \geq \pi^3.
\]

The PMCP 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 specifications ranging from networking to MAS against temporal or epistemic properties [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 \( B = (S, S_0, \Lambda, \mathcal{R}) \), where \( S \) is a finite set of states, \( S_0 \subseteq S \) is a set of initial states, \( \Lambda = L \times \{!!\} \cup \{\} \) and input broadcast labels \( L \times \{!!\} \) and input broadcast labels \( \{\} \), and \( \mathcal{R} \subseteq S \times A \times S \) 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 \( q \) to a global state \( q' \) by means of label \( a \) if there is a process \( i \) such that \( (g.i, a!! \neq q', g'.i) \in R \), and for all processes \( j \neq i \), \( (g.j, a?! \neq q', g.j) \in R \).

**Theorem 1** The PMCP for PCGS is undecidable.

**Proof sketch.** Let \( B = (S, S_0, \Lambda, \mathcal{R}) \) be a broadcast protocol. We construct a PCGS \( \hat{S} = \left( \Sigma, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \right) \) that simulates \( B \) (see Figure 1):

- \( \Sigma_C = \emptyset, \Sigma_T = \{1\} \); \( \hat{Q} = S; \hat{q}_0(1) = S_0; \hat{A} = \Lambda; \hat{\Pi} \) 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 \( a \in \hat{d}(1, q) \).

\( \pi^3 \geq \pi^3 \) is a shortcut for \( \pi \geq \pi^3 \), for each \( 1 \leq i \leq k \).

\( s \)

\[ \begin{array}{c}
  s \quad s^1 \quad s^2 \quad s^3 \quad s^4 \\
  a! \quad s^1 \quad s^2 \quad s^3 \quad s^4 \\
  a?! \quad s^1 \quad s^2 \quad s^3 \quad s^4 \\
  \end{array} \]

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

It follows that for every \( n \in \mathbb{N} \), and for every sequence of states \( \lambda \in (S^n)^+ \), \( \lambda \) is a path in \( B(n) \) iff \( \lambda \) 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 \( \hat{\delta}(1, q, a!!) = q' \). 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 \( \hat{\delta}(aq, q, a, X) = q' \) such that \( a \in X \) for some agent template \( \hat{a} \). 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 \( q \rightarrow_a q' \) to denote the latter, but with \( a \notin X \). That is, if \( q \rightarrow_a q' \), 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 \( \hat{\delta}(aq, q, a, X) = q' \) we have that \( q' \in \hat{q}_0(\hat{a}) \).

**Definition 5 (Resource state)** A template state \( q \) is said to be a resource state if:

1. \( \exists (q', a) \rightarrow_a q \); and
2. \( \exists (q', a) \rightarrow_a q \); and
3. \( q \) 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 \( \hat{a}q \) implies \( q' \rightarrow_a q \) by means of agent template \( \hat{a}q \) and \( q' \rightarrow_a q \) by means of an agent template \( \hat{a}q \) is initialisable.

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.

---

\[ \begin{array}{c}
  s \quad s^1 \quad s^2 \quad s^3 \quad s^4 \\
  a! \quad s^1 \quad s^2 \quad s^3 \quad s^4 \\
  a?! \quad s^1 \quad s^2 \quad s^3 \quad s^4 \\
  \end{array} \]
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 that is 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 $q_2$. 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 $\delta(i, q, a, A)$ bad on action $b$ if $q \neq q'$, $b \in A$, and $\delta(i, q, a, A \setminus \{b\}) \neq q'$. 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 $\delta(1, q_1, a, \{b, c\}, q_2)$ 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 $\delta(i, q, a, A) = q'$ is bad on an action $b$ and $\delta(j, r, b, A) = r'$, for any $j$, $r$, then $r = r'$.

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

Intuitively, the 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 ME class of systems is particularly significant and goes well beyond these protocols. Indeed, in the next section we show that the PMCP for ME MAS against PATL specifications is decidable.

### 4 Verifying the ME Class of PCGS

In this section we show the significance of the ME class of PCGS identified above. Specifically we give a procedure that solve the PMCP for ME PCGS against PATL formulas. Given an ME PCGS and an $\mathcal{m}$-indexed formula, the method involves checking a particular abstract model $S(\mathcal{m})$ against the formula. We show below that the abstract model $S(\mathcal{m})$ 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 $S(\mathcal{m})$, there exists a concrete instantiation of the original PCGS that falsifies the formula.

Fix a PCGS $S = \{\Sigma, Q, q_0, A, \Pi, \pi, d, \delta\}$ in $\mathcal{m}$. Consider the PCGS of one agent template $\hat{\pi}_i$, $\hat{\delta}_i$ and $\hat{\Pi}_i$. The importance of the starvation freedom property.

Let $\varphi$ be an $\mathcal{m}$-indexed PATL formula. For $\mathcal{m} \in \mathbb{N}^+$, let $\Sigma_\varphi(\mathcal{m})$ denote the set $\Sigma_\varphi(\mathcal{m}) = \{(i,j) : 1 \leq i \leq k, 1 \leq j \leq \mathcal{m}\}$ of concrete agents. We now construct the abstract model $\hat{S}(\mathcal{m})$. An abstract state

$$\gamma = \{(q_1, 1), \ldots, (q_k, 1, m), (q_1, 2), \ldots, (q_k, 2), (X_1), \ldots, (X_k)\}$$

consists of a concrete component $\gamma \cdot c = \{(q_1, 1), \ldots, (q_k, 1, m), (q_1, 2), \ldots, (q_k, 2), (X_1), \ldots, (X_k)\}$ and an abstract component $\gamma \cdot ab = \{(X_1), \ldots, (X_k)\}$. The abstract state $\gamma$ represents any concrete state $\varphi$ in an arbitrarily large system $S(\mathcal{m})$ in which: each concrete agent $(i,j) \in \Sigma_\varphi(\mathcal{m})$ is at local state $q_i$; each concrete agent $i \in \Sigma_C$ is at local state $q_i$; for each agent template $i$, $X_i = \{(q_i, j) : (i,j) \in \Sigma_\varphi(\mathcal{m}) \cup \Sigma(\mathcal{m})\}$ is the set of states for the agents not in $\Sigma_\varphi(\mathcal{m})$. We write $\gamma \cdot c \cdot a \cdot b \cdot i$ we denote the abstract component associated with the agent template $i$ in $\gamma$.

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

- $\hat{G} \subseteq \hat{Q}^{\Sigma_\varphi(\mathcal{m})} \times \hat{Q}^{\Sigma(\mathcal{m})} \times \{2^\varphi\} \times \{2^\gamma\}$ is the set of abstract states.
- $\hat{I} = \{\hat{q}_0(1)^{\mathcal{m} \cdot 1} \times \{\hat{q}_0(k)^{\mathcal{m} \cdot k} \times \hat{q}_0(1)^{\mathcal{m} \cdot 1} \times \ldots \times 2^{\mathcal{m} \cdot 1} \times \ldots \times 2^{\mathcal{m} \cdot 1}\}$ is the set of initial abstract states.
- $\hat{T} \subseteq \hat{G} \times \hat{G}$ is the abstract transition relation, where $\Delta = 2^{\hat{Q} \times \hat{I} \times \hat{I}} \times \hat{G}$ is a set of transition labels.
- $\hat{V} : \hat{G} \rightarrow 2^{(\hat{G} \times \hat{I} \times \hat{I}) \cdot (\hat{G} \times \hat{I} \times \hat{I})}$ is the labelling function defined by $(\hat{p}, \hat{q}) \rightarrow \hat{V}(\gamma)(\hat{p}, \hat{\gamma}(\hat{c}, \hat{a}, \hat{b}, \hat{q}))$.

The concrete component $\gamma \cdot c$ encodes the atomic propositions from which $\varphi$ is built, whereas the abstract component $\gamma \cdot ab$ encodes the ways an arbitrary number of agents may interfere with the state of $\gamma \cdot c$. 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 labeled 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 $\Sigma(\mathcal{m}) \cup \Sigma_\varphi(\mathcal{m})$ 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 $q \in S$, $a \in A$, there is exactly one state $q'$ with $(q, a, q') \in R$. 

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

\textbf{Definition 11 (Abstract transition relation)} The abstract transition relation $T$ is defined as \((\Xi_1, \ldots , \Xi_n, \gamma') \in T\) if there is a joint action $\pi \in A(\Sigma(\Xi_1) \cup \cdots \cup \Sigma(\Xi_n))$ such that:

- for every concrete agent $a \gamma = (i, j) \in \Sigma(\Xi)$, we have $\pi.a \in d(i, \gamma, a) \pi$ and $\delta(i, \gamma, a) \pi = \delta(i, \gamma', a) \pi$; $\pi.a \in \text{Actions}(\Xi)$ or $\pi.a = \gamma'$.eq.
- for every concrete agent $i \in C$, we have $\pi.i \in d(i, \gamma, i) \pi$ and $\delta(i, \gamma, i) \pi = \delta(i, \gamma', i) \pi$.
- for each agent template $i$, for all $t = (q, a, A, q') \in \Xi$, we have $q \in \gamma.ab.i$, $q' \in \gamma.ab.i$, $a \in d(i, q), \delta(i, q, a) = q'$, and either $A = \text{Actions}(\pi)$ or $A = \text{Actions}(\pi) \cup \text{Actions}(\Xi \setminus \{i\})$.

- for each agent template $i$, for every $q \in \gamma.ab.i$, there is a transition label in $\Xi$, to a state $q' \in \gamma.ab.i$, and for every $q' \in \gamma.ab.i$, there is a transition label in $\Xi$, from a state $q \in \Xi$, to a state $q'.ab.i$.

A path in $S(\pi)$ is a sequence $\gamma_0 \gamma_1 \ldots$ such that for all $i \geq 0$ there is a k-tuple of transition labels $(\Xi_1, \ldots , \Xi_n)$ with $(\gamma_0, \Xi_1, \ldots, \Xi_n) \in T$. The satisfaction of an $\Xi$-indexed formula $\phi$ on $S(\pi)$ implies the satisfaction of $\phi$ on every concrete system $S(\pi)$ with $\pi \geq \Xi$.

\textbf{Lemma 1} Let $\hat{S} \in \Xi$ be a PCGS of $k$-agent templates and $z$ concrete agents. Let $\phi$ be an $\Xi$-indexed formula. Then, $S(\pi) \models \phi$ implies $S(\pi) \models \phi$ for every $\pi \geq \Xi$.

\textbf{Proof sketch.} Let $\pi \geq \Xi$. Define a mapping $\xi$ from the set of concrete states in $S(\pi)$ to the set of abstract states in $S(\pi)$:

$$
(\xi(q) = (q(1,1), \ldots , q(k, \Xi, k), q_1, \ldots , q_1, \Xi_1, \ldots , \Xi_k),$$

where $\Xi_1 = \{q(1, j) \mid \Xi_1 \cup [\Xi_1 + 1 \leq j \leq \Xi_1]\}$, $\Xi_2 = \{q(k, j) \mid \Xi_2 \cup [\Xi_2 + 1 \leq j \leq \Xi_2]\}$.

For each agent template $i$, define a mapping $\xi_i$ from concrete transitions to abstract transition labels as follows: for a concrete transition $\delta(q, a) = q'$, $(r, a, A', r') \in \xi_i((q, a, (\Xi_1, \ldots , \Xi_k)))$ iff there is a $j$ with $\Xi_1 + 1 \leq j \leq \Xi_1$ such that $\delta(q, a) = (r_i, a_i, r_i')$ and $\delta(q, a) = (r_i, a_i, r_i')$. We do this by induction on $\phi$. Assume $(p, a) \in \Pi(\Xi \times \Sigma(\Xi))$ and $\phi \models (p, a)$. Then, $(p, a) \in \Pi(\gamma, \gamma, a)$, and therefore $(p, a) \in \phi$. Hence, $p = \phi$. The cases of $\phi = \neg \phi_2$, and $\phi_2 \vee \phi_3$ are straightforward.

Suppose that $\phi = \langle \langle \langle \phi_1 \phi_2 \phi_3 \rangle \rangle \rangle$ for a group of agents $\Gamma \subseteq \Sigma(\Xi) \cup \Sigma(\pi)$. Let $\gamma \models \phi$. Consider a strategy $F_\gamma = \{f_{ag} : ag \in \Gamma\}$ such that for all $ag \in \Gamma$, there is $i \geq 0$ with $\phi_1 \models \phi_2$ and $\phi_3 \models \phi_1$ for all $0 \leq j < i$. Choose the strategy for the group of agents $\Gamma$ in $S(\pi)$ such that $\lambda \models \phi$. Let $\lambda = \langle\langle\langle q_0(q_1, q_2, q_3, q_4)\rangle\rangle\rangle$, then $\phi \models q_0(q_1, q_2, q_3, q_4)$. We can check that for each $i \geq 0$, $\phi \models q_0(q_1, q_2, q_3, q_4)$.

Therefore, $\delta(\lambda) = \delta(\phi_0)(d(i, \phi_1, \phi_2, \phi_3))$. Moreover, since $\lambda \models \phi$, it follows that $\delta(\lambda) = \phi$. We can also check that for each $i \geq 0$, $\phi \models q_0(q_1, q_2, q_3, q_4)$.

Finally, $\phi \models \phi = \phi_1 \phi_2$. Hence, $\phi = \phi_1 \phi_2$.

The case of $\phi = \langle \langle \langle \phi \rangle \rangle \rangle \phi_2$ can be shown similarly to the case of $\phi = \langle \langle \langle \phi \rangle \rangle \rangle \phi_2$.
performs the sequences of actions that the agent \( q_i \) performs in \( S(\pi) \); every other agent performs the serial action. Consequently, as the agents in \( \Sigma_C \cup \Sigma_T \) stutter at their initial states in the path from \( q \) to \( q' \), \( S(\pi) \models \varphi \) implies \( S(\pi) \models \varphi \).

\[ \square \]

**Theorem 2** Let \( S \in \mathcal{C} \) be a PCGS and \( \varphi \) an \( \pi\)-indexed formula. Then, the PMCP returns true if and only if \( S(\pi) \models \varphi \).

**Proof** \( (\Rightarrow) \) If \( S(\pi) \models \varphi \), 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 \( \pi\)-indexed PATL formula \( \varphi \), 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 PCGS against any PATL specification: we simply construct the abstract model following Definition 10 and verify it against the specification \( \varphi \). The result of this check is the result of the PMCP for the given PCGS against the PATL \( \varphi \).

### 5 Prioritised Train-Gate-Controller

We exemplify the use of the methodology introduced in this paper in 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 \( \emptyset = \langle \Sigma, Q, \delta_0, A, \Pi, \pi, d, \delta \rangle \), where (see Figure 3):

- \( \Sigma_C = \{ 1 \} \) and \( \Sigma_T = \{ 1, 2 \} \). Template agent \( \hat{1} \) represents the prioritised trains, template agent 2 represents the normal trains, and concrete agent 1 represents the controller.
- \( Q = \{ \emptyset, ?^p, ?^n, \epsilon^p, \epsilon^n, \bot, \top \} \), where:
  - \( \emptyset \) represents a train that 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.
  - \( \epsilon^p \) \((\epsilon^n)\), respectively represents that a prioritised (normal, respectively) train is in the tunnel.
  - \( \bot \) represents the traffic lights having colour red.
  - \( \top \) represents the traffic lights having colour green.
- \( \delta_0 = \{ 1 \mapsto \{ \emptyset \}, \hat{2} \mapsto \{ \emptyset \}, 1 \mapsto \{ \top \} \} \).
- \( A = \{ \operatorname{null}, a_{\bot}^p, a_{\bot}^n, a_{\top}^p, a_{\top}^n, a_{\bot}, a_{\top} \} \), where:
  - \( a_{\bot}^p \) \((a_{\bot}^n)\), respectively represents a prioritised (normal, respectively) train making a request to enter the tunnel.
  - \( a_{\top} \) represents a train relinquishing its request to enter the tunnel.
- \( \hat{A} \) represents the controller accepting a prioritised (normal, respectively) train’s request to enter the tunnel.
- \( \hat{\delta}(\hat{1}, \hat{\varnothing}) = \{ \emptyset, a_{\bot}, a_{\top}^p \} \). 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.
- \( \hat{\delta}(\hat{1}, \hat{\top}) = \{ a_{\bot}^p, a_{\bot}, a_{\top} \} \). 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.
- \( \hat{\delta}(\hat{1}, \hat{\bot}) = \{ a_{\bot}^n \} \). Whenever a train is in the tunnel, it can only exit the tunnel.
- \( \hat{\delta} \) is similarly defined for normal trains.
- \( \hat{\delta}(\hat{1}, \bot) = \hat{\delta}(\hat{1}, \top) = \{ \emptyset \} \).
- \( \hat{\delta}(\hat{1}, ?^p) = \{ \emptyset, a_{\bot}^p \} \), \( \hat{\delta}(\hat{1}, ?^n) = \{ \emptyset, a_{\bot}^n \} \). The controller can either delay a request from a train to enter the tunnel, or it can accept it.
- \( \hat{\delta}(\hat{1}, \varnothing, A_1) = \varnothing \) and \( \hat{\delta}(\hat{1}, \varnothing, A_2, 1) = ?^p \), 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 irrespective of the other agents’ actions. Similarly, \( \hat{\delta}(\hat{1}, ?^p, A_2, 1) = ?^p \), \( \hat{\delta}(\hat{1}, ?^p, a_{\top}, A_1) = \varnothing \), and \( \hat{\delta}(\hat{1}, \epsilon^n, a_{\bot}, A_1) = \varnothing \). These transitions are similarly defined for normal trains.
- \( \hat{\delta}(\hat{1}, ?^p, a_{\bot}, A_2) = \epsilon^n \), where \( \{ a_{\bot}^n \} \subseteq A_2 \subseteq \hat{A}\backslash\{a_{\bot}^n\} \). A prioritised train can enter the tunnel if the controller accepts a prioritised request and no other train is entering the tunnel.

![a.png](attachment:a.png)

Figure 3: The PCGS of 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:

\[
\varphi_{PTGC3} = \forall v, u, z. 2\nu v, 2\nu w, 2G\{\{1\}\} ((tp, v) \rightarrow ((tp, u) \land -(tn, z)) \land (tn, z) \rightarrow (-((tp, u) \land -(tn, w))))
\]

It is easy to check that \( \mathcal{G} \) satisfies the mutual exclusion property and the starvation freedom property. Therefore, \( \mathcal{G} \in \mathcal{ME} \) and therefore we can assess the correctness of the PTGC against the formulae \( \varphi_{PTGCi}, 1 \leq i \leq 3, \) as per Theorem 2. A fragment of the abstract model that represents the fragment of the concrete system \( \mathcal{G}(2, 1) \) (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 omitted in the figure.

The abstract model and the specifications can be checked by an ATL model checker; this would return false for \( \varphi_{PTGC1}, \) and true for \( \varphi_{PTGC2} \) and \( \varphi_{PTGC3}. \) 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 decidability 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.