A Cutoff Technique for the Verification of Parameterised Interpreted Systems with Parameterised Environments

Panagiotis Kouvaros and Alessio Lomuscio
Department of Computing, Imperial College London, UK
{p.kouvaros, a.lomuscio}@imperial.ac.uk

We put forward a cutoff technique for determining the number of agents that is sufficient to consider when checking temporal-epistemic specifications on a system of any size. We identify a special class of interleaved interpreted systems for which we give a parameterised semantics and an abstraction methodology. This enables us to overcome the significant limitations in expressivity present in the state-of-the-art. We present an implementation and discuss experimental results.

1 Introduction
With the development and deployment of autonomous agents and multi-agent systems (MAS) in diverse applications such as search-and-rescue [Murphy, 2000] and web-services [Maximilien and Singh, 2004], there is a growing need to study powerful and versatile techniques for the verification of MAS. A key technique that has emerged in the past ten years is that of model checking [Clarke et al., 1999]. Model checking enables us to check whether a model representing a system satisfies a formula encoding a specification. In the case of MAS a specification expressed by the formula may be not just a temporal formula, as is the case in verification of reactive systems, but a specification given in an agent-based logic, such as BDI [Rao, 1996], Desires-Goal-Intention [Dastani et al., 2003], or temporal-epistemic logic [Fagin et al., 2003].

Indeed, a number of techniques have been put forward for the efficient model checking of MAS against agent-based specifications including binary decision diagrams [Gammie and Meyden, 2004; Lomuscio et al., 2009], abstraction [Cohen et al., 2009], partial order reduction [Lomuscio et al., 2010], bounded model checking [Lomuscio et al., 2007], parallel model checking [Kwiatkowska and A. Lomuscio, 2010], etc., thereby making it possible to verify systems with large state spaces. Yet, since the number of states is exponential in the number of agents in the system, systems of many agents remain intractable. This is particularly problematic when wishing to reason about open MAS. We may be able to encode a system with a given number of agents and verify that a specification holds. But we cannot draw any conclusion as to whether the specification will still hold should more agents be present. We need a technique that enables us to draw conclusions independently on the number of agents present in a MAS.

Cutoffs have been studied in the analysis of parametric systems precisely to solve this problem, often in the context of networking protocols [Emerson and Namjoshi, 1995; Hanna et al., 2009]. A cutoff is the number of components that need to be analysed with respect to a given specification to be able to draw conclusions on the specification in question irrespective on the number of components a system implements. Although the problem of parameterised verification is, in general, undecidable [Apt and Kozen, 1986], sound and incomplete proposals have been put forward [Clarke et al., 2008; German and Sistla, 1992; Hanna et al., 2009; Pnueli et al., 2002; Wolper and Lovinfosse, 1990] that impose restrictions on the systems and the properties studied. In previous work we have begun addressing this in the context of MAS [Kouvaros and Lomuscio, 2013]. However, [Kouvaros and Lomuscio, 2013] makes strong assumptions on the semantics which effectively entail that all agents evolve in the same way following synchronisation with the environment. This is a severe limitation that makes the technique inapplicable to any scenario where agents evolve differently even if they share a single skeleton. The aim of this paper is to overcome these limitations and present a technique which can be employed to verify systems with arbitrary number of agents that do not necessarily evolve in a lock-step fashion.

After recalling the technical setup in Section 2 for interleaved interpreted systems, in Section 3 we develop a semantics for parameterised interpreted systems in which the environments are equipped with parametric actions. This is explored in Section 4 thereby showing how to derive a cutoff for the verification of temporal-epistemic specifications. Section 5 reports on an implementation of the technique and discusses the experimental results obtained for the train-gate-controller scenario with an unbounded number of trains.

2 Model Checking Parameterised Systems
We summarise the formalism of interleaved interpreted systems [Lomuscio et al., 2010] (IIS), a variant of interpreted systems [Fagin et al., 2003] to model asynchronous multi-agent systems, and we discuss parameterised verification in the context of IIS.

Interleaved Interpreted Systems. Consider a MAS composed of $n$ agents $\mathcal{A} = \{1, \ldots, n\}$ and a special agent $E$ (the environment in which the agents “live”). Let $\mathcal{A}' = \mathcal{A} \cup \{E\}$. Each agent $i \in \mathcal{A}'$ is encoded with a nonempty set of local states $L_i$ and a nonempty set of actions Act$_i$, containing a spe-
cational “null” action $\epsilon_i$. For each agent $i \in A'$ assume a local protocol $P_i : L_i \rightarrow \varphi(Act_i)$ governing which actions can be performed in a given state, and a local transition function $t_i : L_i \times Act_i \rightarrow L_i$ specifying the evolution of agent $i$’s local states given its action. The “null” action $\epsilon_i$ is used to model the interleaving evolution of agent $i$; the protocol $P_i$ is such that for every state $l_i \in L_i$ we have that $\epsilon_i \in P_i(l_i)$ (i.e., the null action is enabled at every local state); and the transition function $t_i$ is such that $t_i(l_i, \epsilon_i) = l_i$ (i.e., whenever $\epsilon_i$ is performed, agent $i$’s local state does not change). A global state $g = (l_1, \ldots, l_n, t_E)$ is a tuple of local states for all the agents in the MAS and gives a description of the system at a particular instance of time. Given a global state $g = (l_1, \ldots, l_n, t_E)$ and a set of agents $J = \{j_1, \ldots, j_{|J|}\}$, we write $g_J$ for the tuple of local states $(l_{j_1}, \ldots, l_{j_{|J|}})$ of the agents in $J$ in $g$. We often write $g_i$ instead of $g_{\{i\}}$.

The local protocols and the local evolution functions determine the temporal evolution of the system’s global states. Let $ACT = \bigcup_{i \in A'} Act_i$ and $Agent(a) = \{i \in A' \mid a \in Act_i\}$ for each action $a \in ACT$. The global (interleaved) transition relation $R \subseteq G \times G$ on a set $G$ of global states is defined as $(g, g') \in R$ iff there exists $\pi = (act_1, \ldots, act_n, act_E) \in Act_1 \times \ldots \times Act_n \times Act_E$ and $a \in ACT$ such that for all $i \in Agent(a)$ we have that $act_i = a$ and $t_i(g_i, a) = g_i'$; and for all $i \in A' \setminus Agent(a)$, we have that $act_i = \epsilon_i$ and $t_i(g_i, \epsilon_i) = g_i' = g_i$. We often write $g \xrightarrow{a} g'$ to mean that $(g, g') \in R$ by means of action $a$. Note that only one local action is performed at the system at a given round. Furthermore, every agent potentially able to perform the said action has to perform it at that round. Therefore, the agents communicate by means of shared actions. We assume that the joint null action is always enabled; therefore, $R$ is serial. A path is an infinite sequence $g^1a^1g^2a^2g^3 \ldots$ such that $g^{i+1} = g^{i}a^i$, for every $i \geq 1$. Given a path $\pi$, we write $\pi(i)$ for the $i$-th state in $\pi$. The set of all paths originating from $g$ is denoted by $\Pi(g)$. We write $\pi[i]$ for the suffix $g^{i}a^i g^{i+1} \ldots$ of $\pi$ and $[i] \pi$ for the prefix $g^1a^1 \ldots g^{i} \pi$. A global state $g$ is said to be reachable from a global state $g'$ if there is a path $g'a^i g^{i+1} \ldots$ such that $g' = g^{i}$, for some $i \geq 1$.

**Model Checking IIIs.** Model checking techniques have been developed to verify IIIs against temporal-epistemic specifications [Lomuscio et al., 2010]. However, a key limitation of the approach remains the state explosion problem: the state space of the system grows exponentially in the number of variables encoding the agents. This renders model checking intractable for systems with many agents. Furthermore, since model checking is typically defined for finite state systems, the approach becomes unfeasible for systems with unbounded number of agents, such as open systems.

**Parameterised Systems.** To overcome these limitations, parameterised systems have been put forward [Kouvaros and Lomuscio, 2013]. Parameterised systems are composed of arbitrarily many agents each having identical behaviour. The behaviour of the agents is specified by giving a generic agent template. A parameterised system $S(n)$, where the parameter $n$ is the size of the system, represents an infinite collection of IIIs each composed of $n$ indexed copies of the template agent. In [Kouvaros and Lomuscio, 2013] a cutoff result is established showing that model checking a certain concrete system (for a certain value of the parameter) suffices to establish correctness for all systems. However, the semantics insist on a lock-step evolution of the agents, thereby confining the technique to systems with agents evolving independently of the environment’s actions. The aim of this paper is to overcome these limitations by giving a different semantics and devising a novel verification technique for it.

### 3 Parameterised Environments

We introduce a semantics for parameterised systems in which the environment closely synchronises with the agents’ actions. We use two parameters to describe a MAS: the number of agents in the system and the set of actions of the environment. A Parameterised Interpreted System with Parameterised Environment (or PISPE) is given as follows. Firstly we define a template agent $T = (L, t, Act, P, t')$ and a template environment $E = (L_E, t_E, Act_E, P_E, t_E)$. $T$ and $E$ are encoded as interleaved agents for which we respectively assume an initial state $l \in L$ and $t_E \in L_E$. The set $Act = S \cup NS$ of actions of the template agent is decomposed into disjoint sets of shared actions $S$ and of non-shared actions $NS$. Shared actions are actions shared with the template environment (encoded with these actions; i.e., $Act_E = S$) and they are further decomposed into a set $NPS$ of non-para- 

Parameterised actions and a set $PS$ of parameterised actions.

We assume that non-parameterised actions are shared by all the agents and the environment in a concrete system, whereas parameterised actions are shared by exactly one agent and the environment. Therefore, in compliance with the interleaved semantics, a global transition from a global state $g$ can only happen in three cases: (i) an $NS$ action is enabled for some agent at $g$; (ii) an $NPS$ action is enabled for the environment and all the agents at $g$; (iii) a $PS$ action is enabled for the environment and some agent at $g$. We assume that for each action $a \in S$, the set $\{l \in L_E \mid a \in P_E(l)\}$ is a singleton; i.e., shared actions are enabled by the protocol at exactly one template state in $E$.

Let $n \geq 1$ and $[n] = \{1, \ldots, n\}$. We now describe the construction of a concrete system $S(n)$ from the templates $T$ and $E$. $S(n)$ results from $n$ instantiations $A(n) = \{T(1), \ldots, T(n)\}$, or simply $A(n) = [n]$, of the template agent and an instantiation $E(n)$ of the template environment. Each agent $i \in A(n)$ is instantiated as follows: $L_i = L \times \{i\}$; $Act_i = NS_i \cup PS_i \cup NPS \cup \{\epsilon_i\}$, where $NS_i = NS \times \{i\}$, $PS_i = PS \times \{i\}$; $P_i : L_i \rightarrow \varphi(Act_i)$ is defined by $a \in P_i(l_i)$ iff $t(l) \in P(l)$, where $t(l) = t(x)$, $x \in L_i \cup Act_i$, refers to the corresponding state template or action; $t_i : L_i \times Act_i \rightarrow L_i$ is defined by $t_i(l_i, t_i(a)) = t_i'$ iff $t(l, t(a)) = t'$. The concrete environment $E(n) = (L_E, t_E, Act_E(n), P_E(n), t_E(n))$ is obtained by instantiating each template parameterised action for every concrete agent: $Act_E(n) = NPS \cup PS_E \cup \cdots \cup PS_n$; $P_E(n) : L_E \rightarrow \varphi(Act_E(n))$ is defined by $a \in (P_E(n))(l)$ iff $t(a) \in P_E(l)$; $t_E(n) : L_E \times Act_E(n) \rightarrow L_E$ is defined by $t_E(n)(l, a) = l'$ iff $t_E(l, t(a)) = l'$.

We can now formally define the semantic structures we will be using in this paper.
Definition 3.1 (Parameterised system). Given $T$ and $E$, assume a template labelling function $h : L \rightarrow \phi(AP)$ for a set $AP$ of atomic propositions, and let $n \geq 1$. A parameterised (interleaved) interpreted system with parameterised environment (PISPE, or a model), composed of $n$ concrete agents, is a tuple $\pi(n) = (G(n), \iota(n), R(n), V(n))$, where $G(n) \subseteq L_1 \times \cdots \times L_n \times L_E$ is a set of reachable global states from $\iota(n) = (\iota_1, \ldots, \iota_n, +E)$, $R(n)$ is a global transition relation, and $V(n) : G(n) \rightarrow \phi(AP \times A(n))$ is a labelling function such that $p_i \in (V(n))(g)$ if $p_i \in h(tl(g_i))$.

The above gives a concise description of an infinite collection of concrete systems (or system instances). Each value of the parameter $n$ defines a system composed of a different number of agents and an environment suitable for synchronisation with each of the agents.

Example 3.2. Figure 1 presents a parametric variant of the untimed version of the Train-Gate-Controller (TGC) [Alur et al., 1998] composed of a controller and arbitrarily many trains encoded as a PISPE. Each train runs along a circular track and all tracks pass through a narrow tunnel (template state “T”). The tunnel has enough space for only one train to be in it at any time. The controller operates the colour (template states “G” (Green) and “R” (Red)) of the traffic lights to let the trains enter and exit the tunnel (to state “A” (Away)). Initially, the trains are in state “W” (Waiting) and the controller is in state “G”. In the figure, enter \in \mathbb{P}S and approach \in \mathbb{N}S; the $c$ actions are omitted.

We use the temporal-epistemic logic $\text{ACTL}^+K_{-X}$ [Lomuscio et al., 2010] to expresses properties of MAS to be interpreted on PISPE. The logic combines epistemic modalities with the universal fragment of $\text{CTL}_{-X}$ (the logic $\text{CTL}^*$ without the next-time operator in which: (i) the existential state modality does not appear in any formula; and (ii) the negation is only allowed for atomic propositions). The restriction to a next-step free logic is typical in parameterised verification; the next-step operator can be used to count the number of agents in the system resulting in the verification problem being undecidable [Emerson and Kahl, 2003]. The restriction to universal path quantification is essential in establishing the behavioural equivalence results in Section 4.

Definition 3.3 (Syntax of $\text{ACTL}^+K_{-X}$). State formulae and path formulae of $\text{ACTL}^+K_{-X}$ over a set $AP$ of propositions and a set $A$ of agents are defined by the following BNF ex-

expressions:

\[
\phi ::= p \mid \neg p \mid \phi \land \phi \mid \phi \lor \phi \mid K_i \phi \ (i \in A) \mid A(\psi)
\]

\[
\psi ::= \phi \mid \psi \land \psi \mid \psi \lor \psi \mid U(\psi, \psi) \mid R(\psi, \psi)
\]

where $\phi$ and $\psi$ are state and path formulae and $p \in AP$.

As usual [Fagin et al., 2003] the knowledge modality $K_i$ stands for “agent $i$ knows that”, the path quantifier $A$ is read “for all paths” and the temporal operators $U$, $R$ are read “until” and “release” respectively. $\text{ACTL}^+K_{-X}$ formulae are interpreted in a model $\pi(n)$ as standard: the temporal modalities are interpreted by means of the global transition relation; the epistemic modalities are interpreted over epistemic accessibility relations defined on local equalities for the agents’ states.

Definition 3.4 (Satisfaction). Let $\pi(n)$ be a PISPE, let $\pi = g_1, a_1, g_2, \ldots$ be a path of $\pi(n)$, let $g \in G(n)$ be a state of $\pi(n)$, and let $\phi$ be an $\text{ACTL}^+K_{-X}$ formula. Satisfaction of $\phi$ at $g$, denoted $(\pi(n), g) \models \phi$, or simply $g \models \phi$, and satisfaction of $\phi$ on $\pi$, denoted $(\pi(n), \pi) \models \phi$, or $\pi \models \phi$, is inductively defined as follows:

\[
g = p_i \quad \text{if} \quad p_i \in (V(n))(g);
\]

\[
g = \neg p_i \quad \text{if} \quad g \neq p_i \quad \text{for} \quad p_i \in AP;
\]

\[
g = \phi \land \psi \quad \text{if} \quad g \models \phi \quad \text{and} \quad g \models \psi;
\]

\[
g = \phi \lor \psi \quad \text{if} \quad g \models \phi \quad \text{or} \quad g \models \psi;
\]

\[
g = K_i \phi \quad \text{if} \quad g' \models \phi \quad \text{for} \quad g' \in G(n) \quad \text{such that} \quad g \leadsto g' \quad \text{where} \quad g \sim_i g' \quad \text{if} \quad g_i = g_i';
\]

\[
g = A \phi \quad \text{if} \quad \pi \models \phi \quad \text{for} \quad \text{every path} \pi \quad \text{such that} \quad \pi(1) = g;
\]

\[
\pi \models \phi \quad \text{if} \quad \pi(1) \models \phi \quad \text{for} \quad \text{any state formula} \phi;
\]

\[
\pi \models \phi \land \psi \quad \text{if} \quad \pi \models \phi \quad \text{and} \quad \pi \models \psi;
\]

\[
\pi \models \phi \lor \psi \quad \text{if} \quad \pi \models \phi \quad \text{or} \quad \pi \models \psi;
\]

\[
\pi \models U(\phi, \psi) \quad \text{if} \quad \text{there is an} \quad i \geq 1 \quad \text{such that} \quad \pi[i] \models \psi \\
\text{and} \quad \pi[j] \models \phi \quad \text{for} \quad 1 \leq j < i;
\]

\[
\pi \models R(\phi, \psi) \quad \text{if} \quad \text{for every} \quad i, \quad \pi[j] \neq \phi, \quad \text{for} \quad 1 \leq j < i, \quad \text{then} \quad \pi[i] \models \psi.
\]

We say that a formula $\phi$ is true in a system $\pi(n)$, denoted $\pi(n) \models \phi$, if $(\pi(n), \iota(n)) \models \phi$. We assume the customary abbreviations for $F \phi$ (“Eventually $\phi$”) and $G \phi$ (“Always $\phi$”) from until and release.

We would like to establish whether or not a certain $\text{ACTL}^+K_{-X}$ specification is satisfied by a MAS irrespective of how many agents are present. To do this we consider formulae with atomic propositions and epistemic modalities indexed with variables taking pairwise distinct values in $A(n)$.

We write $\phi(J)$ to indicate that each variable $j \in J$ appears in the formula $\phi$. We verify the PISPE encoding the MAS against properties of the form $\bigwedge_J \phi(J)$. In other words $\bigwedge_J \phi(J)$ is a shortcut for the $\text{ACTL}^+K_{-X}$ formula expressing the conjunction of all $\phi(J)$ under any assignment for $J$.

Note that formulae of this form are implicitly parametric; i.e., the domain $A(n)$ of $J$ depends on the concrete system $\pi(n)$ at which the formula is evaluated. For example consider the property “whenever a train is in the tunnel, it knows that no other train is in the tunnel at the same time” for the TGC. We express this property with the formula $\phi_{TGC} = \bigwedge_{(i,j)} AG(T_i \rightarrow K_i \neg T_j)$ which, when evaluated at $T(2)$, is a shortcut for $AG(T_1 \rightarrow K_1 \neg T_2) \land AG(T_2 \rightarrow K_2 \neg T_1)$. 
Definition 3.5 (Parameterised model checking problem). Given a PISPE \( S(n) \) and an ACTL\(^*\)-K_\text{X} formula \( \phi(J) \), the parameterised model checking problem (PMCP) concerns establishing whether or not the following holds:

\[
\forall n \geq |J| : S(n) \models \bigwedge J \phi(J)
\]

The above amounts to checking an unbounded number of systems. This is a task that in principle involves an unbounded state space which is clearly unfeasible for traditional model checking techniques.

4 MAS Cutoffs

In this section we put forward a cutoff-based abstraction methodology for the PMCP. Cutoffs have been studied to circumvent the aforementioned difficulties in parameterised verification by reducing the number of systems to consider [Emerson and Namjoshi, 1995; Emerson and Kahlon, 2000; Hanna et al., 2009; Siitola, 2010; Kaiser et al., 2010; Kouvares and Lomuscio, 2013]. A cutoff for a system is the number of components that is sufficient to consider when evaluating a given abstraction.

Definition 4.1 (MAS Cutoff). Consider a PISPE \( S(n) \) and let \( \Gamma(I) = \{ \bigwedge J \phi(J) \mid \phi(J) \in \text{ACTL}^*\text{K}_\text{X} \text{ and } J \subseteq I \} \) be a set of specifications for \( S(n) \). A natural number \( c \) is a MAS cutoff for \( S(n) \) with respect to \( \Gamma(I) \) if for any \( \phi \in \Gamma(I) \) we have that \( S(c) \models \phi \iff \forall n \geq c : S(n) \models \phi \).

By definition, if a cutoff exists, then the PMCP can be reduced to model checking all system instances up to the cutoff instance \( S(c) \). Note that by definition a cutoff \( c \) is always greater than the number of agents that any \( \phi \in \Gamma(I) \) can refer to via local propositions and epistemic modalities. Note, also, that \( \Gamma(I) \) contains all ACTL\(^*\)-K_\text{X} formulae with at most \( |I| \) index variables; this is in line with the standard treatment of cutoffs in the literature where a cutoff is defined with respect to the logic under analysis.

Theorem 4.2. There are PISPE \( S(n) \) and specifications \( \Gamma(I) \in \text{ACTL}^*\text{K}_\text{X} \) that admit no cutoff.

Proof. (Sketch) Let \( S(n) \) be the PISPE specified in Figure 2, where \( a, b \in \text{PS} \) and \( d \in \text{NS} \). Suppose that there is a \( c \) such that \( S(c) \models \phi \iff \forall n \geq c : S(n) \models \phi \) for any \( \phi \in \Gamma(I) \). Inductively define a formula \( \delta_c \) as follows: \( \delta_c = s_1 \) if \( c = 1 \); \( \delta_c = \delta_{c-1} \wedge F(u_s \wedge F_{\delta_{c-1}}) \) if \( c > 1 \). Let \( \phi = \bigwedge_{i,j} A(\delta_i \rightarrow G\neg u_j) \) expressing that for each path, if an agent does at least \( c - 1 \) loops through the cycle \( (s, u, s) \), then every other agent never reaches the state \( u \) in the path. It is easy to see that whenever an agent moves to state \( u \), a different agent had moved to state \( t \) in a preceding step. Therefore \( S(c) \models \psi \) and \( S(c+1) \not\models \psi \). The latter contradicts that \( c \) is a cutoff.

While the result above is negative, we now proceed to identify a sufficient condition for the existence of cutoffs.

4.1 Environment Loops

We introduce the notion of shared-simulation between the agent and environment templates. Informally, there is a shared-simulation between \( T \) and \( E \) if \( E \) can simulate \( T \) only by means of the template states in which an \( S \) action is enabled. In the following, let \( l \rightarrow_X l' \), where \( l, l' \) are template states, denote that there exists an action \( a \in X \) such that \( t(l, a) = l' \). Analogously, let \( g \rightarrow_X g' \), where \( g, g' \) are global states, denote that there exists an action \( a \in X \) such that \( g \not\rightarrow_X g' \). Finally, let \( X \rightarrow_X \) denote the reflexive and transitive closure of \( \rightarrow_X \).

Definition 4.3 (Shared-simulation). A relation \( \sim_p \subseteq L \times L_E \) is a shared-simulation between \( T \) and \( E \) if \( l \sim_p l' \) and whenever \( l \sim_p l_E \) the following condition holds: if there is \( l', l'' \in L \) such that \( l \rightarrow_X l'' \rightarrow (a) l'' \) for some \( a \in S \), then \( a \in P_E(l_E) \) and \( l'' \sim_p l_E(a) \).

We write \( T \leq_p E \) to denote that there is a shared-simulation between \( T \) and \( E \). If \( T \leq_p E \), then a looping behaviour is induced on the concrete environment whenever it synchronises between two different agents.

Definition 4.4. A subsequence \( g^1a^1 \ldots g^j \) of a path \( g^1a^1 \ldots \in S(n) \) is an environment loop if \( g^1_E = g^j_E \).

As we clarify below, following synchronisation between the environment and an agent through a PS action, the environment can only synchronise with the same agent unless an environment loop occurs. Therefore, an environment loop occurs whenever the system moves to a state \( g \) in which the environment is able to synchronise with more than one agent.

Definition 4.5. A global state \( \pi(i) \) in a path \( \pi \in S(n) \) has the environment loop condition, denoted \( \text{ELC}(\pi(i), r, q) \), if there exist \( q \neq q \in A(n) \) and \( \pi(j), \pi(j') \), \( j < j' \leq i \), such that \( \pi(j) \rightarrow_{PS} \pi(j') \rightarrow_{NS, \ast} \pi(i) \), and \( \exists g, g' \in G(n) \) such that \( \pi(i) \rightarrow_{PS} g \rightarrow_{\sim_p} g' \).

In other words, a global state \( \pi(i) \) has the environment loop condition if (i) the environment lastly synchronises with an agent \( r \) through a PS action earlier in \( \pi \); and (ii) a different agent \( q \) can asynchronously move from its local state in \( \pi(i) \) to a state in which it can synchronise with the environment through a shared action.

Example 4.6. Consider the path \( \pi = (W_1W_2G) \rightarrow_{enter_1} (T_1W_2R) \rightarrow_{enter_2} (A_1W_2G) \rightarrow_{enter_2} (A_1T_2R) \) of the concrete TGC composed of two trains. The global state \( (A_1, W_2, G) \) in \( \pi \) has the environment loop condition whereas the global state \( (T_1, W_2, R) \) does not.

We call an NP-free section of a path \( \pi \) a subsequence \( g^1 \ldots g^i \) of \( \pi \) such that \( t(l(a^i-1)) \in \text{PS} \) (when \( i > 1 \),

\[ t(a^i-1) \]

\[ \in \text{PS} \]
Lemma 4.7. Suppose that $T \leq \rho \text{ E and let } \rho = g^1a^1, \ldots, g^i$ be an NP-free section of a path in $S(n)$, $n \geq 2$. If $ELC(g^k, r, q), 1 \leq k \leq j$, then $[k] \rho$ is an environment loop.

Proof. (Sketch) By induction on $k$. Base step: $[k] \rho = g^1a^1g^2$. Suppose that $ELC(g^2, r, q)$. We have that $a^1 \notin S_q$ and $\exists g, g^2 : g^2 \rightarrow_{NS} g \rightarrow_{S} g^1$. Obviously, $g^1_E = g_E$ and there is an $a_q \in S_q$ such that $tl(a_q) \in P_E(g_E)$. $T \leq \rho E$ gives that $tl(g^1_E) \sim_p g^1$. Therefore, as $tl(g^2) \rightarrow_{NS} tl(g_0)$, $T \leq \rho E$ gives that $tl(a_q) \in P_E(g^2_E)$. It follows that $g^1_E = g^2_E$. Recall that the set $\{l \in L_E | tl(l) \in P_E(l)\}$ is a singleton which gives that $g^1_E = g^2_E$. Assume for the inductive step that the claim is true for $2 \leq k \leq m - 1$ and suppose that $ELC(g^m, r, q)$. Let $l = \max_{g \in [m-1]}(a^g \in PS_q)$. If such a maximum does not exist, then the claim follows as in the base step; if it exists, then it is easy to see that $ELC(g^1, q, z)$ for some $z \in A(n)$. The inductive hypothesis gives that $g^1_E = g^1_E$. $T \leq \rho E$ gives that $tl(g^1_E) \sim_p g^1$. The latter can be used in a similar argument to the base step and conclude $g^1_E = g^1_E$. Hence, $g^1_E = g^1_E$. □

4.2 Cutoff Theorem

We now show that if there is a shared-simulation between the agent and environment templates, then the cutoff $c$ for a set of specifications $\Gamma(I)$ being considered is $c = \max(2, m)$, where $m = |I|$. We achieve this result by means of three observations. Firstly, we show that by symmetry considerations [Emerson and Sistla, 1996] a formula $\bigwedge_j \phi(J) \in \Gamma(I)$ can be evaluated simply by considering the ground instantiation $\phi([J])$ obtained by assigning the variables in $J$ to any set of distinct values in $A(n)$; for clarity we take the set of values simply to be $\{1, \ldots, |J|\}$.

Lemma 4.8. $\forall n \geq |J| : S(n) \models \bigwedge_j \phi(J) \iff S(n) \models \phi([J])$.

Proof. (Sketch) This follows by suitably extending the result in [Emerson and Sistla, 1996]. □

For example, the formula $\phi_{TAG}$ can be evaluated simply by considering its single conjunct $AG(T_1 \rightarrow K_1\neg T_2)$. Secondly, we show that the cutoff instance $S(c)$ admits the behaviour of any larger system $S(n)$, $n \geq c$. This requires a notion of equivalence between system instances. Recall that ACTL$^K$-X formulae are preserved under stuttering-simulation [Lomuscio et al., 2010]; i.e., if a model $M'$ stutter-simulates a model $M$, denoted $M \leq_{ss} M'$, then $M' \models \phi([m])$ implies that $M \models \phi([m])$. We say that $M \leq_{ss} M'$ if there is a relation $\sim_{ss} \subseteq G \times G'$ such that $t \sim_{ss} t'$ and whenever $g \sim_{ss} g'$ then: (i) if $g \sim g' (i \in [m])$, then $g' \sim g'$ for some $g'$ such that $g^1 \sim g^1$; (ii) $V(g)^{\sim}([p | p \in AP \land m]} = V(g')^{\sim}([p | p \in AP \land g \in [m]}$ and for every $\pi \in \Pi(g)$, there is a $\pi' \in \Pi(g')$, a partition $B_1, B_2, \ldots$ of the states in $\pi$, and a partition $B'_1, B'_2, \ldots$ of the states in $\pi'$ such that for each $j \geq 1$, $B_j$ and $B'_j$ are nonempty and finite, and every state in $B_j$ is related to $\sim_{ss}$ to every state in $B_j'$.

Lemma 4.9. If $T \leq \rho E$ and $S(c) \models \phi([m])$, then $S(n) \models \phi([m])$, for all $n \geq c \geq m$.

Proof. (Sketch) Define a relation $\sim_{ss} = \{(g, g') \in G(n) \times G(c) | g \models \phi([m])$. We show that $S(n) \leq_{ss} S(c)$. Simulation requirement (i): let $g \sim_{ss} g$. Suppose that $g \sim_i g^2$ for some $i \in [m]$ and let $g^2 = g^2_E$. We get that $g^1 \sim g^2$ and $g^2 \sim_{ss} g^2$. Simulation requirement (ii): let $\pi = g^1a^1g^2 \ldots \in \Pi(g')$ and let $\rho = g^1a^1g^2 \ldots$, where $\rho = a^2$ if $a^2 \in \bigcup_{i \in [c]} Act_i$ and $\rho = e$ otherwise. Note that for every subsequence $g^1a^1g^2 \ldots g^1a^1g^2 \ldots$ of an NP-free section in $\pi$ such that $a^1, a^2 \in PS_q$, for some $q \in [c]$, and $a^2 \notin PS_p$, $i < z < j$, we have that $g^1a^1 = g^1E$ by Lemma 4.7. Therefore the environment allows for the transitions in $\rho$, hence $\rho \in \Pi(g')$. Let $B_1, B_2, \ldots$, be a partition of $\pi$ and $\rho$ respectively into singleton blocks. We have that $B_j \sim_{ss} B'_j, j \geq 1$; therefore, $S(n) \sim_{ss} S(c)$. □

Finally, we show that any system $S(n + 1), n \geq c$, admits the behaviour of the system $S(n)$ obtained by removing one component. Repeated application of the following lemma gives that any system $S(n), n \geq c$, simulates the cutoff instance $S(c)$.

Lemma 4.10. If $T \leq \rho E$ and $S(n + 1) \models \phi([m])$, then $S(n) \models \phi([m])$, for all $n \geq c \geq m$.

Proof. (Sketch) The idea is to allow the extra agent $n + 1$ in $S(n + 1)$ to mimic agent 1. Define a relation $R_1 = \{(g, g') \in G(n + 1) | g = g[1] \text{ and } tl(g_{n+1}) \sim_{NS \text{ PS}} tl(g'_1)\};$ if $R_1(g, g'),$ then agent $n + 1$ in $S(n + 1)$ is able to reach the state of agent 1 via PS and NS transitions. We ensure that the environment allows this. Define a relation $R_2 = \{(g, g') \in G(n) \times G(n + 1) | \text{if } g \neq g[1 \text{ then } tl(g_{n+1}) \sim_p g_E, \text{ else } g_E = g_E; \text{ so, if } R_2(g, g') \text{ and there is an NP action enabled at } g, \text{ then } tl(g_{n+1}) \sim_p g_E \text{ thereby allowing agent } n + 1 \text{ to reach the state of agent 1 in } g'. \text{ Now define a relation } \sim_{ss} = R_1 \cap R_2. \text{ We show that } S(n) \leq_{ss} S(n + 1). \text{ Let } g^1 \sim_{ss} g^1$. Simulation requirement (i): follows by a similar argument used in the proof of Lemma 4.9. Simulation requirement (ii): let $\pi = g^1a^1g^2 \ldots \in \Pi(g')$. We inductively construct a path $p = g^1a^1g^2 \ldots \in \Pi(g')$ as required by stuttering-simulation. Assume that we have already constructed a prefix $[j]p$, a prefix $[i]p$ such that $\rho(p) \sim_{ss} \pi(i)$, and a partition of the states in $[j]p$ and $[i]p$ into corresponding blocks. We now define the next blocks $B$ and $B'$. There are two cases depending on the next action $a^1 \in \pi$. Case 1: $tl(a^1) \notin NPS$. Then $B = g^1a^1$ and $B' = g^1a^1$, where $g^1a^1 \rightarrow g^1a^1$. Case 2: $tl(a^1) \in NPS$. Then $B = g^1a^1$ and $B' = g^1a^1g^2a^2 \ldots g^1a^d$, where
\(g^j \rightarrow N S_{c+1} \cup PS_{c+1}; g^{j+1} \rightarrow N S_{c+1} \cup PS_{c+1}; g^{j+1} \rightarrow N S_{c+1} \cup PS_{c+1} \), for each case it can be shown that \(B \sim S_B\); therefore, \(S(n) \leq_{ss} S(n + 1)\).

**Corollary 4.11.** If \(\mathcal{T} \leq_P \mathcal{E}\) then \(\forall n \geq c : S(n) \models \phi \iff S(c) \models \phi\), for any \(\phi \in \Gamma(I)\).

**Proof.** By Lemma 4.8 it suffices to prove the claim for \(\phi([m])\). (\(\Rightarrow\)) Repeated application of Lemma 4.10. (\(\Leftarrow\)) Lemma 4.9.

Following Corollary 4.11 we can reduce the PMCP (Definition 3.5) to the simple check of the cutoff instance \(S(c)\) against \(\phi([m])\).

**5 Implementation and Experimental Results**

We implemented the cut-off based abstraction methodologies presented earlier in an experimental toolkit that we built from the open-source model checker MCMAS [Lomuscio et al., 2009]. We extended ISPL, MCMAS’s input language, to allow for the definition of the semantic structures and the parametric specifications considered here. A PISPE is described by giving declarations for the template agent and the template environment. These extend ISPL’s semantics by considering, among other concepts, the different kind of actions the user can conclude whether or not the specifications hold for any number of agents in the system.

We tested the cutoff technique on the TGC against the specification \(\Delta_{TGC}\), a commonly used benchmark [Lomuscio et al., 2010; Hoek and Wooldridge, 2002]. Note that the state-space grows exponentially with the number of agents in the system. To the best of our knowledge all current techniques would have to consider an unbounded number of systems each with different number of trains. This is unfeasible as illustrated by Table 1: time and space requirements grow exponentially in the number of trains to consider, hence model checking quickly becomes intractable. In our case the base model checker we used could not verify a system composed of 80 trains within the timeout of one hour. In comparison MCMAS-P established the simulation as above and verified the cutoff instance \(S(2)\) in under 0.1 seconds. The MAS cutoff MCMAS-P found corresponds to a MAS with 2 agents; the formula checked, and found to be true, was \(AG(T_1 \rightarrow K_1 \sim T_2)\). This establishes the correctness of the TGC for systems with any number of agents.

**6 Conclusions and Further Work**

As discussed in the Introduction, irrespective of recent progress in the area of verification for MAS, a number of open problems remain, including verification of open systems with an unbounded number of components. Given MAS are often open systems, it seems of particular importance to develop techniques for these setups.

In this paper we put forward a cutoff technique for MAS which enabled us to reason about interleaved interpreted systems in which the agents may synchronise more effectively with the environment. The results we obtained on the shared-simulation relation here defined enabled us to identify a sufficient condition for the cutoff generation. This enabled us to implement a toolkit for the technique which pointed to very significant advantages over conventional model checking. While the results here obtained share the specification language and the general notion of parameterised interpreted system used in [Kouvaros and Lomuscio, 2013], the semantics and the abstraction notion here put forward is different. Most importantly, and differently from [Kouvaros and Lomuscio, 2013], we can here deal with systems in which exactly one concrete agent may synchronise with the environment at a tick of the clock. This enables us to check rich scenarios such as the TGC which are not supported in [Kouvaros and Lomuscio, 2013].

Much work remains to be done in this line. While we can now verify unbounded systems, we can only operate on one abstract template agent. Our future work includes supporting more than one template agent so that we may be able to verify unbounded systems with agents of any kind. This at present remains a considerable challenge due to the complexity of the setup required.

**Acknowledgments.** The research described in this paper was supported by the EPSRC Research Project “Trusted Autonomous Systems” (grant No. EP/I00520X/1).

<table>
<thead>
<tr>
<th>#Trains</th>
<th>#States</th>
<th>Time (s)</th>
<th>Memory (KiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.15 \times 10^{27}</td>
<td>3</td>
<td>10005</td>
</tr>
<tr>
<td>40</td>
<td>2.30 \times 10^{13}</td>
<td>128</td>
<td>47792</td>
</tr>
<tr>
<td>60</td>
<td>3.57 \times 10^{19}</td>
<td>1317</td>
<td>60998</td>
</tr>
</tbody>
</table>

**Table 1:** Verification results for the TGC.
References


