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 M_S representing a system S, satisfies a formula ϕ_P encoding a specification P. In the case of MAS a specification expressed by the formula ϕ_P 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 $\mathcal{A} = \{1, \dots, n\}$ 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 special "null" action ϵ_i. For each agent $i \in \mathcal{A}'$ assume a local protocol $P_i : L_i \to \wp(Act_i)$ governing which actions can be performed in a given state, and a local transition function $t_i : L_i \times Act_i \to L_i$ specifying the evolution of agent i's local states given its action. The "null" action ϵ_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 ϵ_i is performed, agent i's local state does not change). A *global state* $g = (l_1, \ldots, l_n, l_E)$ 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 $g = (l_1, \ldots, l_n, l_E)$ 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 $ACT = \bigcup_{i \in \mathcal{A}'} Act_i$ and $Agent(a) = \{i \in \mathcal{A}' \mid a \in Act_i\}$ for each action $a \in ACT$. The global (interleaved) transition relation $\mathcal{R} \subseteq G \times G$ on a set G of global states is defined as $(g, g') \in \mathcal{R}$ iff there exists $\bar{a} = (act_1, \ldots, act_n, act_E) \in Act_1 \times \ldots \times Act_n \times Act_E$ 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 $i \in \mathcal{A}' \setminus Agent(a)$, we have that $act_i = \epsilon_i$ and $t_i(g_i, act_i) = g_i' = g_i$. We often write $g \xrightarrow{a} g'$ to mean that $(g, g') \in \mathcal{R}$ by means of action a. Note that only one local action is performed in 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, $\mathcal{R}$ is serial. A path is an infinite sequence $g^1 a^1 g^2 a^2 g^3 \ldots$ such that $g^i \xrightarrow{a^i} g^{i+1}$, for every $i \geq 1$. Given a path π, we write $\pi(i)$ for the i-th state in π. 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 π and $[i]\pi$ for the prefix $g^1 a^1 \ldots g^i$ of π. A global state g is said to be reachable from a global state g^1 if there is a path $g^1 a^1 g^2 \ldots$ such that $g = g^i$, for some $i \geq 1$.

Model Checking IIS. Model checking techniques have been developed to verify IIS 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.

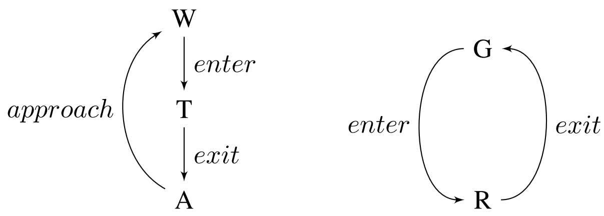
Parameterised Systems. To overcome these limitations, parameterised systems have been put forward [Kouvaros and Lomuscio, 2013]. Parameterised systems are composed of arbitrarily many agents each having identical behaviour. The behaviour of the agents is specified by giving a generic agent *template*. A parameterised system $\mathcal{S}(n)$, where the parameter n is the size of the system, represents an infinite collection of IIS 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 $\mathcal{T} = \langle L, \iota, Act, P, t\rangle$ and a template environment $\mathcal{E} = \langle L_E, \iota_E, Act_E, P_E, t_E\rangle$. $\mathcal{T}$ and $\mathcal{E}$ are encoded as interleaved agents for which we respectively assume an initial state $\iota \in L$ and $\iota_E \in L_E$. 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-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 $\mathcal{E}$.

Let $n \geq 1$ and $[n] = \{1, \ldots, n\}$. We now describe the construction of a concrete system $\mathcal{S}(n)$ from the templates $\mathcal{T}$ and $\mathcal{E}$. $\mathcal{S}(n)$ results from n instantiations $\mathcal{A}(n) = \{\mathcal{T}(1), \ldots, \mathcal{T}(n)\}$, or simply $\mathcal{A}(n) = [n]$, of the template agent and an instantiation $\mathcal{E}(n)$ of the template environment. Each agent $i \in \mathcal{A}(n)$ is instantiated as follows: $L_i = L \times \{i\}$; $Act_i = NS_i \cup PS_i \cup NPS \cup \{\epsilon_i\}$, where $NS_i = NS \times \{i\}$, $PS_i = PS \times \{i\}$; $P_i : L_i \to \wp(Act_i)$ is defined by $a \in P_i(l_i)$ iff $tl(a) \in P(l)$, where $tl(x), x \in L_i \cup Act_i$, refers to the corresponding template state or action; $t_i : L_i \times Act_i \to L_i$ is defined by $t_i(l_i, a) = l_i'$ iff $t(l, tl(a)) = l'$. The concrete environment $\mathcal{E}(n) = \langle L_E, \iota_E, Act_E(n), P_E(n), t_E(n)\rangle$ is obtained by instantiating each template parameterised action for every concrete agent: $Act_E(n) = NPS \cup PS_1 \cup \cdots \cup PS_n$; $P_E(n) : L_E \to \wp(Act_E(n))$ is defined by $a \in (P_E(n))(l)$ iff $tl(a) \in P_E(l)$; $t_E(n) : L_E \times Act_E(n) \to L_E$ is defined by $(t_E(n))(l, a) = l'$ iff $t_E(l, tl(a)) = l'$.

We can now formally define the semantic structures we will be using in this paper.



(a) Template Agent (Train)

(b) Template Environment (Controller)

Figure 1: A PISPE of parameterised TGC.

Definition 3.1 (Parameterised system)**.** *Given $\mathcal{T}$ and $\mathcal{E}$, assume a template labelling function $h : L \rightarrow \wp(AP)$ 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 $\mathcal{S}(n) = \langle G(n), \iota(n), \mathcal{R}(n), V(n) \rangle$, where $G(n) \subseteq L_1 \times \cdots \times L_n \times L_E$ is a set of reachable global states from $\iota(n) = (\iota_1, \ldots, \iota_n, \iota_E)$, $\mathcal{R}(n)$ is a global transition relation, and $V(n) : G(n) \rightarrow \wp(AP \times \mathcal{A}(n))$ is a labelling function such that $p_i \in (V(n))(g)$ iff $p \in h(tl(g_i))$.*

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 $enter, exit \in PS$ and $approach \in NS$; the ϵ actions are omitted.*

We use the temporal-epistemic logic 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 CTL$^*_{-X}$ (the logic 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 Kahlon, 2003]. The restriction to universal path quantification is essential in establishing the behavioural equivalence results in Section 4.

Definition 3.3 (Syntax of ACTL*K$_{-X}$)**.** *State formulae and path formulae of ACTL*K$_{-X}$ over a set AP of propositions and a set $\mathcal{A}$ of agents are defined by the following BNF expressions:*

$$\phi ::= p \mid \neg p \mid \phi \wedge \phi \mid \phi \vee \phi \mid K_i \phi \ (i \in \mathcal{A}) \mid A(\psi)$$
$$\psi ::= \phi \mid \psi \wedge \psi \mid \psi \vee \psi \mid U(\psi, \psi) \mid R(\psi, \psi)$$

where ϕ and ψ 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. ACTL*K$_{-X}$ formulae are interpreted in a model $\mathcal{S}(n)$ 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 $\mathcal{S}(n)$ be a PISPE, let $\pi = g^1, a^1, g^2, \ldots$ be a path of $\mathcal{S}(n)$, let $g \in G(n)$ be a state of $\mathcal{S}(n)$, and let ϕ be an ACTL*K$_{-X}$ formula. Satisfaction of ϕ at g, denoted $(\mathcal{S}(n), g) \models \phi$, or simply $g \models \phi$, and satisfaction of ϕ on π, denoted $(\mathcal{S}(n), \pi) \models \phi$, or $\pi \models \phi$, is inductively defined as follows:*

$g \models p_i$	if	$p_i \in (V(n))(g)$;
$g \models \neg p_i$	if	not $g \models p_i$, for $p \in AP$;
$g \models \phi \wedge \psi$	if	$g \models \phi$ and $g \models \psi$;
$g \models \phi \vee \psi$	if	$g \models \phi$ or $g \models \psi$;
$g \models K_i \phi$	if	$g' \models \phi$ for every $g' \in G(n)$ such that $g \sim_i g'$, where $g \sim_i g'$ if $g_i = g'_i$;
$g \models A\phi$	if	$\pi \models \phi$ for every path π such that $\pi(1) = g$;
$\pi \models \phi$	if	$\pi(1) \models \phi$ for any state formula ϕ;
$\pi \models \phi \wedge \psi$	if	$\pi \models \phi$ and $\pi \models \psi$;
$\pi \models \phi \vee \psi$	if	$\pi \models \phi$ or $\pi \models \psi$;
$\pi \models U(\phi, \psi)$	if	there is an $i \geq 1$ such that $\pi[i] \models \psi$ and $\pi[j] \models \phi$ for all $1 \leq j < i$;
$\pi \models R(\phi, \psi)$	if	for every i, if $\pi[j] \not\models \phi$, for all $1 \leq j < i$, then $\pi[i] \models \psi$.

We say that a formula ϕ is true in a system $\mathcal{S}(n)$, denoted $\mathcal{S}(n) \models \phi$, if $(\mathcal{S}(n), \iota(n)) \models \phi$. We assume the customary abbreviations for $F\phi$ ("Eventually ϕ") and $G\phi$ ("Always ϕ") from until and release.

We would like to establish whether or not a certain 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 $\mathcal{A}(n)$. We write $\phi(J)$ to indicate that each variable $j \in J$ appears in the formula ϕ. We verify the PISPE encoding the MAS against properties of the form $\bigwedge_J \varphi(J)$. In other words $\bigwedge_J \phi(J)$ is a shortcut for the 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 $\mathcal{A}(n)$ of J depends on the concrete system $\mathcal{S}(n)$ 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 $\mathcal{S}(2)$, is a shortcut for $AG(T_1 \rightarrow K_1 \neg T_2) \wedge AG(T_2 \rightarrow K_2 \neg T_1)$.

Similar properties, but specified in CTL*, are considered in some approaches to parameterised verification [Clarke *et al.*, 1989].

Definition 3.5 (Parameterised model checking problem)**.** *Given a PISPE $\mathcal{S}(n)$ and an ACTL*K$_{-X}$ formula $\phi(J)$, the parameterised model checking problem (PMCP) concerns establishing whether or not the following holds:*

$$\forall n \geq |J| : \mathcal{S}(n) \models \bigwedge_J \phi(J)$$

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; Siirtola, 2010; Kaiser *et al.*, 2010; Kouvaros and Lomuscio, 2013]. A cutoff for a system is the number of components that is sufficient to consider when evaluating a given specification.

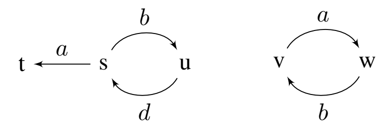
Definition 4.1 (MAS Cutoff)**.** *Consider a PISPE $\mathcal{S}(n)$ and let $\Gamma(I) = \{\bigwedge_J \phi(J) \mid \phi(J) \in$ ACTL*K$_{-X}$ and $J \subseteq I\}$ be a set of specifications for $\mathcal{S}(n)$. A natural number c is a* MAS cutoff *for $\mathcal{S}(n)$ with respect to $\Gamma(I)$ if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

By definition, if a cutoff exists, then the PMCP can be reduced to model checking all system instances up to the cutoff instance $\mathcal{S}(c)$. 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$_{-X}$ 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 $\mathcal{S}(n)$ and specifications $\Gamma(I) \in$ ACTL*K$_{-X}$ that admit no cutoff.*

Proof. (Sketch) Let $\mathcal{S}(n)$ be the PISPE specified in Figure 2, where $a, b \in PS$ and $d \in NS$. Suppose that there is a c such that $\mathcal{S}(c) \models \phi \Leftrightarrow \forall n \geq c : \mathcal{S}(n) \models \phi$, for any $\phi \in \Gamma(\{i, j\})$. Inductively define a formula δ_c as follows: $\delta_c = s_i$ if $c = 1$; $\delta_c = s_i \wedge F(u_i \wedge F\delta_{c-1})$ if $c > 1$. Let $\phi = \bigwedge_{\{i,j\}} A(\delta_c \rightarrow G\neg u_j)$ 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 $\mathcal{S}(c) \models \psi$ and $\mathcal{S}(c+1) \not\models \psi$. The latter contradicts that c is cutoff. $\square$

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



(a) Template Agent (b) Template Environment

Figure 2: A PISPE with no cutoff.

4.1 Environment Loops

We introduce the notion of *shared-simulation* between the agent and environment templates. Informally, there is a shared-simulation between $\mathcal{T}$ and $\mathcal{E}$ if $\mathcal{E}$ can simulate $\mathcal{T}$ 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 $g \xrightarrow{a} g'$. Finally, let $\rightarrow_{X*}$ 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 $\mathcal{T}$ and $\mathcal{E}$ if $\iota \sim_P \iota_E$ and whenever $l \sim_P l_E$ the following condition holds: if there is $l', l'' \in L$ such that $l \rightarrow_{NS*} l' \rightarrow_{\{a\}} l''$ for some $a \in S$, then $a \in P_E(l_E)$ and $l'' \sim_P t_E(l_E, a)$.*

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

Definition 4.4. *A subsequence $g^i a^i \ldots g^j$, $j > i$, of a path $g^1 a^1 \ldots$ in $\mathcal{S}(n)$ is an* environment loop *if $g^i_E = g^j_E$.*

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 π in $\mathcal{S}(n)$ has the* environment loop condition*, denoted $ELC(\pi(i), r, q)$, if there exist $r \neq q \in \mathcal{A}(n)$ and $\pi(j), \pi(j')$, $j < j' \leq i$, such that $\pi(j) \rightarrow_{PS_r} \pi(j') \rightarrow_{NS_r*} \pi(i)$, and $\exists g, g' \in G(n)$: $\pi(i) \rightarrow_{NS_q*} g \rightarrow_{S_q} g'$.*

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 π; 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 $\pi = (W_1 W_2 G) \xrightarrow{enter_1} (T_1 W_2 R) \xrightarrow{exit_1} (A_1 W_2 G) \xrightarrow{enter_2} (A_1 T_2 R)$ of the concrete TGC composed of two trains. The global state (A_1, W_2, G) in π 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 π a subsequence $g^i a^i \ldots g^j$ of π such that $tl(a^{i-1}) \in NPS$ (when $i > 1$),

$tl(a^k) \notin NPS$ for $i \leq k \leq j-1$ and $tl(a^j) \in NPS$. In other words the agents either perform asynchronous actions, or they synchronise with the environment only. The following lemma shows that the environment loop condition is a sufficient (but not necessary) condition for the occurrence of an environment loop in an NP-free section. Specifically, whenever $ELC(g, r, q)$ holds, the environment's local state in g is equal to its local state at the initial global state of the NP-free section in which g occurs.

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

Proof. (Sketch) By induction on k. Base step: $[k]\rho = g^1 a^1 g^2$. Suppose that $ELC(g^2, r, q)$. We have that $a^1 \notin S_q$ and $\exists g, g' : g^2 \to_{NS_q*} g \to_{S_q} g'$. Obviously, $g_E^2 = g_E$ and there is an $a_q \in S_q$ such that $tl(a_q) \in P_E(g_E)$. $\mathcal{T} \leq_P \mathcal{E}$ gives that $tl(g_q^1) \sim_P g_E^1$. Therefore, as $tl(g_q^1) \to_{NS_q*} tl(g_q)$, $\mathcal{T} \leq_P \mathcal{E}$ gives that $tl(a_q) \in P_E(g_E^1)$. It follows that $g_E^1 = g_E$ (recall that the set $\{l \in L_E \mid tl(a_q) \in P_E(l)\}$ is a singleton) which gives that $g_E^1 = g_E^2$. 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 $l = \max_{y \in [m-1]}(a^y \in PS_q)$. 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 $ELC(g^{l+1}, q, z)$ for some $z \in \mathcal{A}(n)$. The inductive hypothesis gives that $g_E^1 = g_E^{l+1}$; $\mathcal{T} \leq_P \mathcal{E}$ gives that $tl(g_q^{l+1}) \sim_P g_E^{l+1}$. The latter can be used in a similar argument to the base step and conclude $g_E^{l+1} = g_E^m$. Hence, $g_E^1 = g_E^m$. $\square$

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 $\mathcal{A}(n)$; for clarity we take the set of values simply to be $\{1, \ldots, |J|\}$.

Lemma 4.8. $\forall n \geq |J| : \mathcal{S}(n) \models \bigwedge_J \phi(J)$ *iff* $\mathcal{S}(n) \models \phi([|J|])$.

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

For example, the formula ϕ_{TGC} can be evaluated simply by considering its single conjunct $AG(T_1 \to K_1 \neg T_2)$.

Secondly, we show that the cutoff instance $\mathcal{S}(c)$ admits the behaviour of any larger system $\mathcal{S}(n)$, $n \geq c$. 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 $\mathcal{M}'$ stuttering-simulates a model $\mathcal{M}$, denoted $\mathcal{M} \leq_{ss} \mathcal{M}'$, then $\mathcal{M}' \models \phi([m])$ implies that $\mathcal{M} \models \phi([m])$. We say that $\mathcal{M} \leq_{ss} \mathcal{M}'$ if there is a relation $\sim_{ss} \subseteq G \times G'$ such that $\iota \sim_{ss} \iota'$ and whenever $g \sim_{ss} g'$ then: (i) if $g \sim_i g^1$ ($i \in [m]$), then $g' \sim_i g'^1$ for some g'^1 such that $g^1 \sim_{ss} g'^1$; (ii) $V(g) \cap \{p_i \mid p \in AP \wedge i \in [m]\} = V'(g') \cap \{p_i \mid p \in AP \wedge i \in [m]\}$ 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 π, and a partition $B'_1, B'_2, \ldots$ of the states in π' such that for each $j \geq 1$, B_j and B'_j are nonempty and finite, and every state in B_j is related by $\sim_{ss}$ to every state in B'_j.

Lemma 4.9. *If $\mathcal{T} \leq_P \mathcal{E}$ and $\mathcal{S}(c) \models \phi([m])$, then $\mathcal{S}(n) \models \phi([m])$, for all $n \geq c \geq m$.*

Proof. (Sketch) Define a relation $\sim_{ss} = \{(g, g') \in G(n) \times G(c) \mid g_{[c]} = g'\}$. We show that $\mathcal{S}(n) \leq_{ss} \mathcal{S}(c)$. Simulation requirement (i): let $g^1 \sim_{ss} g'^1$. Suppose that $g^1 \sim_i g^2$ for some $i \in [m]$ and let $g'^2 = g^2_{[c]}$. We get that $g'^1 \sim_i g'^2$ and $g^2 \sim_{ss} g'^2$. Simulation requirement (ii): let $\pi = g^1 a^1 g^2 \ldots \in \Pi(g^1)$ and let $\rho = g^1_{[c]} a'^1 g^2_{[c]} \ldots$, where $a'^j = a^j$ if $a^j \in \bigcup_{i \in [c]} Act_i$ and $a'^j = \epsilon$ otherwise. Note that for every subsequence $g^i a^i g^{i+1} \ldots g^j a^j g^{j+1}$ of an NP-free section in π such that $a^i, a^j \in PS_q$, for some $q \in [c]$, and $a^z \notin PS_q$, $i < z < j$, we have that $g_E^{i+1} = g_E^j$ by Lemma 4.7. Therefore the environment allows for the transitions in ρ, hence $\rho \in \Pi(g'^1)$. Let $B_1, B_2, \ldots$ and $B'_1, B'_2, \ldots$ be a partition of π and ρ respectively into singleton blocks. We have that $B_j \sim_{ss} B'_j$, $j \geq 1$; therefore, $\mathcal{S}(n) \leq_{ss} \mathcal{S}(c)$. $\square$

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

Lemma 4.10. *If $\mathcal{T} \leq_P \mathcal{E}$ and $\mathcal{S}(n+1) \models \phi([m])$, then $\mathcal{S}(n) \models \phi([m])$, for all $n \geq c \geq m$.*

Proof. (Sketch) The idea is to allow the extra agent $n+1$ in $\mathcal{S}(n+1)$ to mimic agent 1. Define a relation $R_1 = \{(g, g') \in G(n) \times G(n+1) \mid g = g'_{[n]}$ and $tl(g'_{n+1}) \to_{(NS \cup PS)*} tl(g'_1)\}$; if $R_1(g, g')$, then agent $n+1$ in $\mathcal{S}(n+1)$ is able to reach the state of agent 1 via PS and NS transitions. We ensure that the environment allows this. Define a relation $R_2 = \{(g, g') \in G(n) \times G(n+1) \mid$ if $\exists g^1 : g \to_{NPS} g^1$, then $tl(g'_{n+1}) \sim_P g'_E$, else $g'_E = g_E\}$; so, if $R_2(g, g')$ and there is an NPS action enabled at g, then $tl(g'_{n+1}) \sim_P g'_E$ thereby allowing agent $n+1$ to reach the state of agent 1 in g'. Now define a relation $\sim_{ss} = R_1 \cap R_2$. We show that $\mathcal{S}(n) \leq_{ss} \mathcal{S}(n+1)$. 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 $\pi = g^1 a^1 g^2 \cdots \in \Pi(g^1)$. We inductively construct a path $\rho = g'^1 a'^1 g'^2 \cdots \in \Pi(g'^1)$ as required by stuttering-simulation. Assume that we have already constructed a prefix $[j]\rho$, a prefix $[i]\pi$ such that $\rho(j) \sim_{ss} \pi(i)$, and a partition of the states in $[j]\rho$ and $[i]\pi$ into corresponding blocks. We now define the next blocks B and B'. There are two cases depending on the next action a^i in π. Case 1: $tl(a^i) \notin NPS$. Then $B = g^{i+1}$ and $B' = g'^{j+1}$, where $g'^j \xrightarrow{a^i} g'^{j+1}$. Case 2: $tl(a^i) \in NPS$. Then $B = g^{i+1}$ and $B' = g'^{j+1} g'^{j+2} \ldots g'^{j+d}$, where

$g'^j \rightarrow_{NS_{n+1} \cup PS_{n+1}} g'^{j+1} \cdots \rightarrow_{NS_{n+1} \cup PS_{n+1}} g'^{d-1} \xrightarrow{a^i} g'^d$. For each case it can be shown that $B \sim_{ss} B'$; therefore, $\mathcal{S}(n) \leq_{ss} \mathcal{S}(n+1)$. □

Corollary 4.11. *If $\mathcal{T} \leq_P \mathcal{E}$ then $\forall n \geq c : \mathcal{S}(n) \models \phi$ iff $\mathcal{S}(c) \models \phi$, 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 $\mathcal{S}(c)$ 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 that PISPE are defined on. We refer to [MCMAS-P, 2013] for more details.

Given the input descriptions for $\mathcal{T}$ and $\mathcal{E}$, the model checker MCMAS-P first attempts to establish whether $\mathcal{T} \leq_P \mathcal{E}$. To do this, the tool constructs the concrete system $\mathcal{S}(1)$. The states in $\mathcal{S}(1)$ are assigned atomic propositions by the valuation function $V(1) : G(1) \rightarrow \wp(AP)$, where $AP = \{a_{\mathcal{T}}, a_{\mathcal{E}} \mid a \in S\}$, defined as $a_{\mathcal{T}} \in (V(1))(g)$ iff $a \in P(tl(g_1))$ and $a_{\mathcal{E}} \in (V(1))(g)$ iff $a \in P_E(g_E)$. In other words a state g is labelled with $a_{\mathcal{T}}$ (respectively, $a_{\mathcal{E}}$) if the shared action a is enabled for the agent (respectively, the environment) at g. Following this, MCMAS is called to check $\mathcal{S}(1)$ against the formulae in $\Delta = \{AG(a_{\mathcal{T}} \rightarrow a_{\mathcal{E}}) \mid a \in S\}$. It is easy to see that $\mathcal{T} \leq_P \mathcal{E}$ iff $\forall \delta \in \Delta : \mathcal{S}(1) \models \delta$; so if the formulae are satisfied we can conclude that $\mathcal{T} \leq_P \mathcal{E}$. Following a successful simulation test MCMAS-P employs Corollary 4.11 by means of five steps: (i) the cutoff c is computed from the cardinality of the set of indices used in the specifications $\Lambda \subset \Gamma(I)$ to check; (ii) the reachable state-space of the cutoff system $\mathcal{S}(c)$ is computed and encoded symbolically; (iii) given an (input) valuation of the template states, atomic propositions are assigned to global states as in Definition 3.1; (iv) the specification formulae are reduced to their ground instantiations $\Lambda([c])$ as in Lemma 4.8; (v) finally, MCMAS is called to verify $\mathcal{S}(c)$ against $\Lambda([c])$. Following these calculations 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 ϕ_{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 $\mathcal{S}(2)$ 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 $AG(T_1 \rightarrow K_1 \neg T_2)$. This establishes the correctness of the TGC for systems with any number of agents.

#Trains	#States	Time (s)	Memory (KiB)
2 (Cutoff)	**8**	**0**	**8774**
20	1.15×10^{27}	3	10005
40	2.30×10^{13}	128	47792
60	3.57×10^{19}	1317	60998
80	TIMEOUT	TIMEOUT	TIMEOUT

Table 1: Verification results for the TGC.

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).

References

[Alur *et al.*, 1998] R. Alur, TA. Henzinger, FYC. Mang, S. Qadeer, SK.—Rajamani, and S. Tasiran. MOCHA: User Manual. In *cMocha (Version 1.0. 1) Documentation*, 1998.

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

[Clarke *et al.*, 1989] 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.

[Clarke *et al.*, 1999] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 1999.

[Clarke *et al.*, 2008] E.M. Clarke, M. Talupur, and H. Veith. Proving ptolemy right: The environment abstraction framework for model checking concurrent systems. In *Proceedings of TACAS'08*, pages 33–47. Springer, 2008.

[Cohen *et al.*, 2009] M. Cohen, M. Dam, A. Lomuscio, and F. Russo. Abstraction in model checking multi-agent systems. In *Proceedings of AAMAS'09*, pages 945–952. IFAAMAS Press, 2009.

[Dastani *et al.*, 2003] M. Dastani, M. van Riemsdijk, F. Dignum, and J. .J.Meyer. A programming language for cognitive agents goal directed 3APL. In *Proceedings of ProMAS'03*, pages 111–130. Springer, 2003.

[Emerson and Kahlon, 2000] E. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *Proceedings of CADE'00*, pages 236–254. Springer, 2000.

[Emerson and Kahlon, 2003] E.A. Emerson and V. Kahlon. Model checking guarded protocols. In *Proceedings of LICS'03*, pages 361–370. IEEE, 2003.

[Emerson and Namjoshi, 1995] E.A. Emerson and K.S. Namjoshi. Reasoning about rings. In *Proceedings POPL'95*, pages 85–94. Pearson Education, 1995.

[Emerson and Sistla, 1996] E.A. Emerson and A.P. Sistla. Symmetry and model checking. *Formal methods in system design*, 9(1):105–131, 1996.

[Fagin *et al.*, 2003] R. Fagin, Y. Moses, J. Y. Halpern, and M. Y. Vardi. *Reasoning about knowledge*. The MIT Press, 2003.

[Gammie and Meyden, 2004] P. Gammie and R. Van Der Meyden. Mck: Model checking the logic of knowledge. In *Proceedings of CAV'04*, pages 256–259. Springer, 2004.

[German and Sistla, 1992] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM (JACM)*, 39(3):675–735, 1992.

[Hanna *et al.*, 2009] Y. Hanna, S. Basu, and H. Rajan. Behavioral automata composition for automatic topology independent verification of parameterized systems. In *Proceedings of ESEC/FSE'09*, pages 325–334. ACM, 2009.

[Hoek and Wooldridge, 2002] W. Van Der Hoek and M. Wooldridge. Tractable multi-agent planning for epistemic goals. In *Proceedings of AAMAS'02*, pages 1167–1174. IFAAMAS, 2002.

[Kaiser *et al.*, 2010] A. Kaiser, D. Kroening, and T. Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *Proceedings of CAV'10*, pages 645–659. Springer, 2010.

[Kouvaros and Lomuscio, 2013] P. Kouvaros and A. Lomuscio. Automatic verification of parameterised interleaved multi-agent systems. In *Proceedings of AAMAS'13*. IFAAMAS, 2013. To Appear.

[Kwiatkowska and A. Lomuscio, 2010] M. Kwiatkowska and H. Qu A. Lomuscio. Parallel model checking for temporal epistemic logic. In *Proceedings of ECAI'10*, pages 543–548. IOS Press, 2010.

[Lomuscio *et al.*, 2007] A. Lomuscio, W. Penczek, and B. Woźna. Bounded model checking knowledge and real time. *Artificial Intelligence*, 171(16-17):1011–1038, 2007.

[Lomuscio *et al.*, 2009] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of CAV'09*, pages 682–688. Springer, 2009.

[Lomuscio *et al.*, 2010] A. Lomuscio, W. Penczek, and H. Qu. Partial order reductions for model checking temporal-epistemic logics over interleaved multi-agent systems. *Fundamenta Informaticae*, 101(1):71–90, 2010.

[Maximilien and Singh, 2004] E. M. Maximilien and M. P. Singh. A framework and ontology for dynamic web services selection. *Internet Computing*, 8(5):84–93, 2004.

[MCMAS-P, 2013] MCMAS-P. Model Checking Parameterised Multi-Agent Systems. `http://vas.doc.ic.ac.uk/software/tools`, 2013.

[Murphy, 2000] R. R. Murphy. Marsupial and shape-shifting robots for urban search and rescue. *Intelligent Systems and their Applications*, 15(2):14–19, 2000.

[Pnueli *et al.*, 2002] A. Pnueli, J. Xu, and L. Zuck. Liveness with (0, 1,infinity)-counter abstraction. In *Proceedings of CAV'02*, pages 93–111. Springer, 2002.

[Rao, 1996] A. Rao. Agentspeak (L): BDI agents speak out in a logical computable language. In *Proceedings of MAAMAW'96*, pages 42–55. Springer, 1996.

[Siirtola, 2010] A. Siirtola. Automated multiparameterised verification by cut-offs. *Formal Methods and Software Engineering*, 6447:321–337, 2010.

[Wolper and Lovinfosse, 1990] P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In *Proceedings of AVMFSS'89*, pages 68–80. Springer, 1990.