Automatic Verification of Parameterised Interleaved Multi-Agent Systems

Panagiotis Kouvaros
Department of Computing
Imperial College London
panagiotis.kouvaros10@imperial.ac.uk

Alessio Lomuscio
Department of Computing
Imperial College London
a.lomuscio@imperial.ac.uk

ABSTRACT

A key problem in verification of multi-agent systems by model checking concerns the fact that the state-space of the system grows exponentially with the number of agents present. This often makes practical model checking unfeasible whenever the system contains more than a few agents. In this paper we put forward a technique to establish a cutoff result, thereby showing that systems with an arbitrary number of agents can be verified by checking a single system consisting of a number of agents equal to the cutoff of the system. While this problem is undecidable in general, we here define a class of parameterised interpreted systems and a parameterised temporal-epistemic logic for which the result can be shown. We exemplify the theoretical results on a robotic example and present an implementation of the technique as an extension of MCMAS, an open-source model checker for multi-agent systems.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model Checking

General Terms

Theory; Verification

Keywords

Epistemic Logic; Model Checking; Parameterised MAS

1. INTRODUCTION

Verification and validation of systems before deployment is increasingly seen of fundamental importance not just in safety-critical applications, but also in more mainstream applications. Multi-Agent Systems (MAS) are no exception. The past ten years have witnessed considerable research in verification techniques aimed at assessing automatically whether or not a MAS meets its intended specifications.

One of the leading techniques in this area is *model checking* [5]. In this setting the system S under analysis is encoded as a transition system M_S and a specification P is formalised as a logical formula ϕ_P; a model checking procedure is used to determine whether $M_S \models \phi_P$, i.e., whether or not the system M_S satisfies the formula ϕ_P. Since MAS specifications are often expressed as epistemic, deontic, and ATL formulas; the techniques put forward in the MAS community reflect this [15, 13, 17]. While explicit techniques are less efficient, symbolic checkers such as MCK [13], MCMAS [17] and VERICS [15] are capable of handling state-spaces of the region of 10^{15} and beyond. However, MAS-based applications, due to the agents' complex and intentional nature, often generate much larger state spaces. To alleviate this problem, a number of techniques, including abstraction [7], have been put forward. These have been successful in allowing users to tackle larger systems, but it is still the case that, generally speaking, systems with many agents are difficult to verify. The aim of this paper is to contribute to overcome this shortcoming.

In the present setting we consider a class of MAS composed of identical agents interacting with the environment. While this may seem a strong condition, this is a relatively common assumption in many application areas of interest ranging from robotics, to artificial life, swarm intelligence, services and in open systems in general. A natural question in these systems is whether certain properties hold *irrespective of the number of agents present*. For example, in a remote robotic scenario we may wish to check that a goal is met irrespective of how many robots are present. It is immediate to see that plain model checking cannot be used to solve this problem. To establish this property we would have to consider an infinite number of different systems each composed of a different number of agents and run model checking algorithms on each of these. Not withstanding the fact that we cannot check an infinite number of systems, this class includes instances for which model checking would require an unfeasible amount of memory and time.

In this paper we develop a technique that enables us to derive the number of agents that is sufficient to consider to show that a property holds in the system *for any number of agents*. In line with the literature on reactive systems [8, 14], we call this bound the *MAS cutoff*. In contrast with literature in reactive systems we here work with *interleaved interpreted systems* [16] and temporal-epistemic specifications.

The rest of the paper is organised as follows. In Section 2 we define an interleaved semantics that will use throughout the paper, a logic that we call IACTL*K$_{-X}$ which combines the universal fragment of CTL* without "next" with a parameterised version of epistemic logic, and establish a stuttering-equivalence simulation result. Section 3 introduces the technique to establish the cutoff and presents our main theoretical result. To exemplify the theory we discuss a robotic example in Section 4. We discuss our implementation

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013),* Ito, Jonker, Gini, and Shehory (eds.), May 6–10, 2013, Saint Paul, Minnesota, USA.

in Section 5 and present experimental results. We conclude in Section 6 also discussing related work.

2. PARAMETERISED INTERLEAVED INTERPRETED SYSTEMS

In this section we introduce a framework for reasoning about parameterised multi-agent systems. In particular, we recall the semantics of interleaved interpreted systems [16] and we introduce parameterised interleaved multi-agent systems. To reason about the temporal-epistemic properties of agents, we introduce the logic IACTL*K$_{-X}$, a parameterised extension of ACTL$^*_{-X}$ with indexed atomic propositions and indexed epistemic modalities.

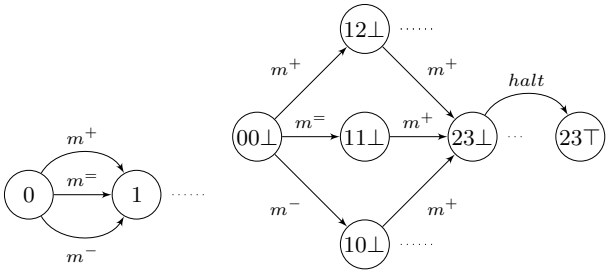
2.1 Interleaved Interpreted Systems

The interpreted systems (IS) formalism [12] is a standard semantics for MAS. Here we consider a special class of interpreted systems, called interleaved interpreted systems (IIS) [16], in which the agents evolve in parallel asynchronously (i.e., by means of interleaving semantics [5]). Differently from standard interpreted systems where actions may be performed by all the agents at the same round, IIS insist on only one local action at the time to be performed in the system. If at any given round more than one agent admits in its repertoire the action to be performed, then all agents sharing this action perform this action at that round. Thus, the agents communicate by means of shared actions. The temporal evolution of an agent's local states is accommodated to the needs of interleaving; while in standard IS the next local state depends on the actions performed by all agents in the system, in IIS local states depend only on the agent's own action. Below, we summarise the framework of IIS, as presented in [16], to model interleaved MAS.

We assume that a MAS is composed of n agents $\mathcal{A} = \{1, \dots n\}$. Each agent $i \in \mathcal{A}$ is characterised by a finite set of local states L_i and a finite set of actions Act_i. Each Act_i contains a special action ϵ_i which we call the "silent" action; as the name suggests, whenever ϵ_i is performed, agent i's local state does not change. We call $ACT = \bigcup_{i\in\mathcal{A}} Act_i$ the union of all actions. Actions are performed in compliance with a protocol $P_i : L_i \to \wp(Act_i)$ governing which actions can be executed in a given state. The silent action is enabled at every local state; formally, $\forall i \in \mathcal{A} : \forall l_i \in L_i : \epsilon_i \in P_i(l_i)$. For each action a, we call $Agent(a) = \{i \in \mathcal{A} \mid a \in Act_i\}$ the set of agents potentially able to perform a. The evolution of agent i's local states is described by the transition function $t_i : L_i \times Act_i \to L_i$ such that $t_i(l_i, \epsilon_i) = l_i$ for each $l_i \in L_i$. Note that t_i is a function of agent i's local action only.

A global state $g = (l_1, \cdots, l_n) \in L_1 \times \cdots \times L_n$ is an n-tuple of local states for all the agents in the MAS and represents the state of the system at a particular instance of time. Given a global state $g = (l_1, \dots, l_n)$, we write g_i to denote the local component l_i of agent $i \in \mathcal{A}$ in g. Given a set of agents $J = \{j_1, \dots, j_{|J|}\} \subseteq \mathcal{A}$, we write g_J to denote the tuple of local components $(l_{j_1}, \dots, l_{j_{|J|}})$ of agents J in g. The local protocols and the local evolution functions determine how the system proceeds from one global state to the next.

Definition 2.1. *(Interleaved Semantics) Let G be a set of global states. The global interleaved evolution function $t : G \times Act_i \times \cdots \times Act_n \to G$ is defined as follows: $t(g, a_1, \dots, a_n) = g'$ iff there exists an action $a \in ACT$ such that for all $i \in Agent(a)$ we have that $a_i = a$ and $t_i(g_i, a) = g'_i$; and for all $i \in \mathcal{A} \backslash Agent(a)$, we have that $a_i = \epsilon_i$ and $t_i(g_i, a_i) = g'_i = g_i$. In short, we write the above as $g \xrightarrow{a} g'$.*



(a) Environment (b) Robot

Figure 1: The IIS for the Autonomous Robot Example.

We assume that the joint silent action is always enabled; thus, the global transition relation is serial. A sequence of global states and actions $\pi = g^1 a^1 g^2 a^2 g^3 \dots$ is said to be an interleaved path (or simply a path) originating at g^1 if for every pair of successor states we have that $g^i \xrightarrow{a^i} g^{i+1}$, for every $i \geq 1$. We write $\pi(i)$ to denote the i-th global state in π. The set of all paths originating from g is denoted by $\Pi(g)$. The local path of agent $i \in \mathcal{A}$ in π is the projection of π onto i; i.e., the sequence $\pi^i = g^1_i a^1 g^2_i a^2 g^3_i \dots$. The projection of π onto a set of agents J is the sequence $\pi^J = g^1_J a^1 g^2_J a^2 g^3_J \dots$. We denote by $\pi[i]$ the suffix $g^i a^i g^{i+1} \cdots$ of π. A state $g \in G$ is said to be reachable from $g^1 \in G$ if there is a path $g^1 a^1 g^2 \cdots$ such that $g = g^i$, for some $i \geq 1$.

Definition 2.2. *(Interleaved Interpreted Systems) Let AP be a set of atomic propositions. An* interleaved interpreted system *(IIS), or a model, is a 4-tuple $\mathcal{M} = \langle G, \iota, \Pi, V \rangle$, where G is a set of global states, $\iota \in G$ is an initial global state such that each state in G is reachable from ι, $\Pi = \bigcup_{g\in G} \Pi(g)$ is the set of all interleaved paths originating from all states in G, and $V : AP \to \wp(G)$ is a valuation function.*

Example 2.3. *The IIS presented in Figure 1 is a modified version of the autonomous robot (AR) example from [12]. A robot runs along an endless straight track; its position is given in terms of locations numbered as $0, 1, 2, \cdots$. The robot can only move forward along the track starting at position 0. A faulty sensor is attached to the robot measuring its position; a sensor reading at location q can be any of the values in $R_q = \{q-1, q, q+1\}$. The movement of the robot is controlled by the environment. The only action the robot can perform is to halt; if the robot does not halt, the environment may move the robot one position forward at each time step; once the robot halts, the environment can no longer move it. The goal of the robot is never to exit the goal region $GR = \{2, 3, 4\}$ upon entering into it and never to halt in the restricted region $RR = \{0, 1\}$. A sound and complete solution to the autonomous robot problem [12] is for the robot to do nothing while the value of its sensor is less than 3 and to halt once the value of its sensor is greater than or equal to 3. In the figure a state of the environment represents the position of the robot, and a state psh of the robot represents, respectively, its position, its sensor reading, and whether it has halted or not.*

2.2 Template Agent and Parameterised Interleaved Systems

Several protocols are designed for an unbounded number of identical participants. Cache coherence, mutual exclusion, and voting protocols are typical examples in which the number of participants (caches, processes, and voters respectively) is independent of the design process. Multi-party negotiation protocols, auctions and open MAS in general also have this property. In the following we develop a formal model of a parameterised multi-agent system, composed of an arbitrary number of identical agents, that can be used in these circumstances. Given that the number of agents is a priori unknown, a parameterised system describes an infinite family of systems where an instance in the family, or concrete instantiation, is obtained by specifying the number of agents in the system. Formally, we introduce below parameterised interleaved interpreted systems (PIIS), an extension of interleaved interpreted systems, to model the aforementioned classes of systems.

We write $\mathcal{T}(n)$ to denote a PIIS, where $n \geq 1$ is the parameter specifying the number of agents, each constructed from a template agent $\mathcal{T}$. The template agent is an interleaved agent encoded with a set of synchronous actions and a set of asynchronous actions. As it will be clear below, if the action performed in a global transition is a synchronous action, then all agents participate in the global action by performing the same synchronous action. However, if the action performed in a global transition is an asynchronous action, then exactly one agent participates in the global action. Therefore, all agents synchronise at any time step in which a synchronous action is performed.

Definition 2.4. *(Template Agent) Given a set of propositions AP, a* template agent *is a tuple $\mathcal{T} = \langle L, \iota, Act, P, t, h\rangle$, where L is a finite nonempty set of template states from which $\iota \in L$ is the unique initial template state, $Act = Act^S \cup Act^A \cup \epsilon$ is a finite set of template actions, where Act^S is a set of synchronous actions, Act^A is a set of asynchronous actions with $Act^S \cap Act^A \cap \{\epsilon\} = \emptyset$, $P : L \to \wp(Act)$ is the protocol such that for all $l \in L$, $\epsilon \in P(l)$, $t : L \times Act \to L$ is the deterministic template evolution function such that for all $l \in L$, $t(l, \epsilon) = l$, and $h : L \to \wp(AP)$ is a labelling function for the template states.*

Given a template agent $\mathcal{T}$, $\mathcal{T}(n)$ denotes the parallel composition of n concrete agents[1] in $\mathcal{A} = \{1, \cdots, n\}$. Each agent $i \in \mathcal{A}$ is obtained by subscripting the states and actions of $\mathcal{T}$ as follows: $L_i = L \times \{i\}$, $Act_i = Act^S \cup Act_i^A \cup \epsilon_i$, where $Act_i^A = Act^A \times \{i\}$; synchronous template actions are not subscripted. For a concrete action $a \in Act_i$, we write $tl(a)$ to refer to the corresponding template action; analogously, for a concrete state $l_i \in L_i$, we write $tl(l_i)$ to refer to the corresponding template state l. The local protocol $P_i : L_i \to \wp(Act_i)$ of the i-th agent is defined by $a \in P_i(l_i)$ iff $tl(a) \in P(l)$. The evolution function $t_i : L_i \times Act_i \to L_i$ of the i-th agent is defined by $t_i(l_i, a) = l_i'$ iff $t(l, tl(a)) = l'$. We associate with each $i \in \mathcal{A}$ a local labelling function $V_i : L_i \to \wp(AP \times \{i\})$ defined by $p_i \in V_i(l_i)$ iff $p \in h(l)$.

The global transitions we consider in PIIS are as in Definition 2.1. A global transition from a global state g complies with the definition if either a synchronous action is enabled for all agents in g or an asynchronous action $a_i \in Act_i$ is enabled for an agent $i \in \mathcal{A}$ in g. Indeed, if $a \in Act^S$, then $Agent(a) = \mathcal{A}$; if $a_i \in Act_i^A$, for some $i \in \mathcal{A}$, then $Agent(a) = \{i\}$. Therefore, synchronous actions play the role of shared actions in IIS, but here synchronous actions are shared by all agents. We now define parameterised interleaved interpreted systems.

Definition 2.5. *(Parameterised Interleaved Interpreted Systems) Given a natural number $n \geq 1$ and a template agent $\mathcal{T} = \langle L, \iota, Act, P, t, h\rangle$, a* parameterised interleaved interpreted system *(PIIS), composed of n concrete agents, is a tuple $\mathcal{T}(n) = \langle G^n, \iota^n, \Pi^n, V^n\rangle$, where $G^n = L \times [n]$ is a set of global states, $\iota^n = (\iota_1, \ldots, \iota_n)$ is an initial (global) state, $\Pi^n = \bigcup_{g \in G^n} \Pi(g)$ is the set of all interleaved paths originating from all states in G^n, and $V^n : G^n \to \wp(AP \times \mathcal{A})$ is a labelling function defined by $p_i \in V^n(g)$ iff $p_i \in V_i(g(i))$.*

Given a template agent, the above definition denotes an infinite family of concrete systems. A member of the family, which we call an instance of the parameterised system, is obtained by fixing the value of the parameter n.

2.3 The Specification Language IACTL*K$_{-\mathbf{X}}$

Temporal-epistemic logic has been widely adopted to express the properties of agents in a MAS. However, we cannot use propositional temporal-epistemic logics to reason about an unbounded number of agents. To see this, consider the parameterised variant of the autonomous robot and suppose that we want to express the property: "for every $i \in \mathcal{A}$, whenever i halts, then it knows that every other robot's position is within the goal region". This property encodes all distinct pairs of robots. Therefore, to express the property for an AR composed of n robots we need to construct a formula composed of $2!\binom{n}{2}$ conjuncts. Instead we would like to express properties that are independent of the number of agents in the system, as if we were able to quantify over the agents. To overcome these shortcomings we introduce the indexed temporal-epistemic logic IACTL*K$_{-X}$. Indexed logics are commonly used in parameterised systems [8].

IACTL*K$_{-X}$ combines indexed epistemic modalities with the universal fragment of CTL$^*_{-X}$ [4] (the logic CTL*, without the next-time operator, extended with indexed atomic propositions). We consider a stuttering-insensitive logic, i.e., a logic insensitive to repeated occurrences of the same state, or equivalently a logic without the next-time operator [5]). This is because the next-time operator can be used to count the number of agents in the system [4, 9] resulting in the parameterised verification problem being undecidable [9].

Intuitively, any IACTL*K$_{-X}$ formula φ represents an ACTL*K$_{-X}$ formula for each concrete system $\mathcal{T}(n)$, $n \geq c$, where c is the number of unique indices contained in φ; the formula corresponding to $\mathcal{T}(n)$ is the conjunction of all formulae that can be constructed from φ by instantiating the indices with every c-tuple of distinct agents in $\mathcal{T}(n)$.

2.3.1 Syntax and semantics of IACTL*K$_{-X}$

We assume a set VS of variable symbols which we use to index the atomic propositions and the epistemic modalities. There are two types of formulas in IACTL*K$_{-X}$: (i) state formulas which are true at a state and (ii) path formulas which are true on a path.

[1]When it is clear from the context, we write "agent" instead of "concrete agent".

DEFINITION 2.6. *(Syntax of IACTL$^*K_{-X}$) The state and path formulae of IACTL$^*K_{-X}$ over a set AP of propositions and a set VS of variable symbols are inductively defined as follows:*

- *S1. if $p \in AP$ and $v \in VS$, then p_v and $\neg p_v$ are state formulas;*
- *S2. if φ and ψ are state formulas, then $\varphi \wedge \psi$, $\varphi \vee \psi$ and $K_v\varphi$ ($v \in VS$) are state formulas;*
- *S3. if φ is a state formula with exactly $J \subseteq VS$ variable symbols, then $\bigwedge_J \varphi$ is a state formula;*
- *S4. if φ is a path formula, then $A(\varphi)$ is a state formula;*
- *P1. any state formula φ is also a path formula;*
- *P2. if φ and ψ are path formulas, then $\varphi \wedge \psi$ and $\varphi \vee \psi$ are path formulas;*
- *P3. if φ and ψ are path formulas, then $U(\varphi, \psi)$ and $R(\varphi, \psi)$ are path formulas.*

The $\bigwedge_J$ connective serves as a universal agent quantifier ranging over all $|J|$-tuples of pairwise distinct agents. Given a formula φ, a variable $v \in VS$, occurring in φ, is said to be bound if it is in the scope of a $\bigwedge_J$ connective; otherwise, v is said to be free. A formula in which there are no free occurrences of variables is said to be a sentence. We here consider only sentences. For an IACTL*K$_{-X}$ formula φ, we write $\varphi(J)$ to indicate that: (i) all variables in $J \subseteq VS$ and only them occur free in φ, and (ii) φ does not contain any $\bigwedge_J$ connectives. The path quantifier A stands for "for all paths". The temporal operators U and R represent "until" and "release" respectively; the formula $U(\varphi, \psi)$ is read as "φ holds continuously until ψ holds", whereas the formula $R(\varphi, \psi)$ is read as "φ releases ψ". The operator K denotes the epistemic modality; $K_v\varphi$ is read as "each concrete agent $i \in \mathcal{A}$ knows φ". Since we consider sentences only, v is always bound by a $\bigwedge_J$ connective; therefore v ranges over all agents.

Consider the AR again; we can now easily express the properties previously stated by considering the IACTL*K$_{-X}$ formula $\varphi_{AR1} = \bigwedge_{\{i,j\}} AG\,(h_i \rightarrow K_i g_j)$, where the atomic propositions h_i, g_i hold respectively in the states where robot i has halted and is within the goal region.

The specifications we consider in this paper are of the form $\bigwedge_J \varphi(J)$. Since these formulas range over all $|J|$-tuples of distinct agents, a standard model checking procedure would have to consider every instantiation of $\varphi(J)$. However, a result we obtain is that model checking a formula $\bigwedge_J \varphi(J) \in$ IACTL*K$_{-X}$ *can be reduced to model checking a single instantiation of* $\varphi(J)$, thereby simplifying the complexity of the model checking procedure. Note that an instantiation of $\varphi(J)$ is an ACTL*K$_{-X}$ formula, built as follows.

DEFINITION 2.7. *ACTL$^*K_{-X}$ formulae over a set AP of atomic propositions and a set $\mathcal{A}$ of agents are defined as in Definition 2.6 but omitting (S3) and replacing (S1) and (S2) with: S1'. if $p \in AP$ and $i \in \mathcal{A}$, then p_i and $\neg p_i$ are state formulas; S2'. if φ and ψ are state formulas, then $\varphi \wedge \psi$, $\varphi \vee \psi$ and $K_i\varphi$ ($i \in \mathcal{A}$) are state formulas;*

For an ACTL*K$_{-X}$ formula φ, we write $\varphi(J)$ ($J \subseteq \mathcal{A}$) to indicate that for each subformula $K_i\psi$ and each proposition p_j of φ we have that $i, j \in J$. We write ACTL*K$^J_{-X}$ for the restriction of ACTL*K$_{-X}$ to all formulae of the form $\varphi(J)$. We interpret IACTL*K$_{-X}$ formulae over PIIS. The temporal modalities are interpreted over the global transition relation and the epistemic modalities are interpreted over the equality of the local components of the global states.

DEFINITION 2.8. *(Satisfaction) Let $\mathcal{T}(n) = \langle G, \iota, \Pi, V\rangle$ be a parameterised interleaved interpreted system, let $\pi = g^1, a^1, g^2, \ldots$ be a path of $\mathcal{T}(n)$, let $g \in G$ be a state of $\mathcal{T}(n)$, and let φ be an IACTL$^*K_{-X}$ formula. Satisfaction of φ at g, denoted $(\mathcal{T}(n), g) \models \varphi$, or simply $g \models \varphi$, and satisfaction of φ on π, denoted $(\mathcal{T}(n), \pi) \models \varphi$, or just $\pi \models \varphi$, is inductively defined as follows:*

S1.	$g \models p_i$	iff	$p_i \in V_i(g_i)$;
	$g \models \neg p_i$	iff	not $g \models p_i$, for $p_i \in AP \times \mathcal{A}$;
S2.	$g \models \varphi \wedge \psi$	iff	$g \models \varphi$ and $g \models \psi$;
	$g \models \varphi \vee \psi$	iff	$g \models \varphi$ or $g \models \psi$;
	$g \models K_i\varphi$	iff	$g' \models \varphi$ for every $g' \in G$ such that $g_i = g'_i$;
S3.	$g \models \bigwedge_J \varphi(J)$	iff	$g \models \varphi(C)$ for every $C \in \{I \mid I \subseteq \mathcal{A}$ and $\lvert I\rvert = \lvert J\rvert\}$;
S4.	$g \models A\varphi$	iff	$\pi \models \varphi$ for every path π such that $\pi(1) = g$;
P1.	$\pi \models \varphi$	iff	$\pi(1) \models \varphi$ for any state formula φ;
P2	$\pi \models \varphi \wedge \psi$	iff	$\pi \models \varphi$ and $\pi \models \psi$;
	$\pi \models \varphi \vee \psi$	iff	$\pi \models \varphi$ or $\pi \models \psi$;
P3	$\pi \models U(\varphi, \psi)$	iff	there is an $i \geq 1$ such that $\pi[i] \models \psi$ and $\pi[j] \models \varphi$ for all $1 \leq j < i$;
	$\pi \models R(\varphi, \psi)$	iff	for every i, if $\pi[j] \not\models \varphi$, for all $1 \leq j < i$, then $\pi[i] \models \psi$.

We use the following abbreviations: $\top \stackrel{def}{\equiv} p_v \vee \neg p_v$, $\bot \stackrel{def}{\equiv} p_v \wedge \neg p_v$, for some $p \in AP$ and $v \in VS$, $F\varphi \stackrel{def}{\equiv} U(\top, \varphi)$ ("Eventually φ"), $G\varphi \stackrel{def}{\equiv} R(\bot, \varphi)$ ("Always φ"). A formula φ is said to be true in $\mathcal{T}(n)$, denoted $\mathcal{T}(n) \models \varphi$, iff $(\mathcal{T}(n), \iota) \models \varphi$.

2.3.2 Symmetry reduction for ACTL$^*K_{-X}$

Symmetry reduction techniques have been used to reduce the complexity of model checking temporal-epistemic properties of multi-agent systems [6]. Since a PIIS is composed of identical agents, intuitively, there is an inherent symmetry in the system that we can exploit. Indeed, we adapt a result from reactive systems [11] and we show that an IACTL*K$_{-X}$ formula $\bigwedge_J \varphi(J)$ is equivalent to a single instantiation $\varphi(\{1, \ldots, |J|\})$ of $\varphi(J)$.

LEMMA 2.9. *$\mathcal{T}(n) \models \bigwedge_J \varphi(J)$ iff $\mathcal{T}(n) \models \varphi(\{1, \ldots, |J|\})$.*

PROOF. *(Sketch)* ($\Rightarrow$) Obvious. ($\Leftarrow$) Suppose that $\mathcal{T}(n) \models \varphi(\{1, \cdots, k\})$ and let $J' = \{j_1, \cdots, j_k\}$ be an arbitrary set of k agents. Let $\zeta : \mathcal{A} \rightarrow \mathcal{A}$ be a bijective mapping such that $\forall i \in \{1, \cdots, k\} : \pi(i) = j_i$. Given an object o (either a state or a formula), let $\zeta(o)$ denote the object o' obtained by replacing each occurrence of any $i \in \mathcal{A}$ with $\zeta(i)$. As $g \stackrel{a}{\rightarrow} g'$ is iff $\zeta(g) \stackrel{\zeta(a)}{\rightarrow} \zeta(g')$, and $g_i = g'_i$ is iff $(\zeta(g))_{\zeta(i)} = (\zeta(g'))_{\zeta(i)}$, we get that $(\mathcal{T}(n), \zeta(\iota)) \models \zeta(\varphi(\{1, \cdots, k\})$, and therefore $(\mathcal{T}(n), \zeta(\iota)) \models \varphi(J')$. As $\zeta(\iota) = \iota$, we get that $(\mathcal{T}(n), \iota) \models \varphi(J)$, therefore $\mathcal{T}(n) \models \bigwedge_J \varphi(J)$. $\square$

Given the semantics of IACTL*K$_{-X}$, model checking a formula $\bigwedge_J \varphi(J)$ on a system $\mathcal{T}(n)$ is equivalent to model checking an ACTL*K$_{-X}$ formula of $c!\binom{c}{|J|}$ conjuncts of the form $\varphi(C)$, where $C \in \{I \mid I \subseteq \{1, \cdots, c\} \text{ and } |I| = |J|\}$. By the above lemma we can model check a single conjunct. For example, model checking the formula $\bigwedge_{\{i,j\}} AG\,(h_i \rightarrow K_i g_j)$ can be reduced to model checking the simpler formula $AG(h_1 \rightarrow K_1 g_2)$.

2.3.3 Invariance of ACTL$^*K_{-X}$

As noted above, cutoff techniques concern the identification of a system instance, called the cutoff instance, which can be used to check whether a given property holds for all system instances. A notion of equivalence between the system instances is often used to show this result. Stuttering-insensitive logics are accompanied with the standard notion of stuttering simulation [18]. A system stuttering simulates another system if for every behaviour of the latter, there is a stuttering equivalent behaviour of the former. Informally, two behaviours are stuttering equivalent if the behaviours coincide when each sequence of stutter steps (i.e., steps that do not affect the labelling of the states) are collapsed onto a single step. Since we consider only universal path quantification, it follows that any ACTL$^*{}_{-X}$ formula satisfied by the simulating model is also satisfied by the simulated model. Below we extend the notion of stuttering equivalence to ACTL*K$_{-X}$. Since our specifications are of the form $\varphi(J)$, referring only to agents in $J \subseteq \mathcal{A}$ via atomic propositions, we project the valuation function onto J. This projection, denoted $V|_J$, is defined by $V|_J(g) = V(g) \cap \{p_i \mid i \in J\}$, for every state $g \in G$.

Definition 2.10. *(J-Stuttering simulation) A relation $\sim_{Jss} \subseteq G \times G'$ is a J-stuttering simulation between two models $\mathcal{M} = \langle G, \iota, \Pi, V\rangle$ and $\mathcal{M}' = \langle G', \iota', \Pi', V'\rangle$ if the following conditions hold:*

1. *$\iota \sim_{Jss} \iota'$;*
2. *if $g^1 \sim_{Jss} g'^1$ then*
   (a) *if $g_i^1 = g_i^2$, for $i \in J$, then $g_i'^1 = g_i'^2$ for some g'^2 such that $g^2 \sim_{Jss} g'^2$;*
   (b) *$V|_J(g^1) = V|_J(g'^1)$ and for every path π of $\mathcal{M}$ that starts at g^1, there is a path π' of $\mathcal{M}'$ that starts at g'^1, a partition $B_1, B_2 \ldots$ of π, and a partition $B'_1, B'_2, \ldots$ of 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 $\sim_{Jss}$ to every state in B'_j.*

A model $\mathcal{M}'$ J-stuttering simulates a model $\mathcal{M}$, denoted $\mathcal{M} \leq_{Jss} \mathcal{M}'$, if there is a J-stuttering simulation between $\mathcal{M}$ and $\mathcal{M}'$. Two models $\mathcal{M}$ and $\mathcal{M}'$ are called J-stuttering simulation equivalent if $\mathcal{M} \leq_{Jss} \mathcal{M}'$ and $\mathcal{M}' \leq_{Jss} \mathcal{M}$. ACTL*K$^J_{-X}$ formulae are preserved under J-stuttering simulation equivalence.

Theorem 2.11. *Let $\mathcal{M}$ and $\mathcal{M}'$ be two J-stuttering simulation equivalent models. Then, $(\mathcal{M}, \iota) \models \varphi$ iff $(\mathcal{M}', \iota') \models \varphi$, for any ACTL$^*K^J_{-X}$ formula φ.*

Proof. Stuttering simulation equivalence is known to preserve ACTL$^*_{-X}$ formulae [18]. Since atomic propositions in ACTL$^{*J}_{-X}$ formulae refer only to agents in $J \subseteq \mathcal{A}$, it follows that J-stuttering simulation equivalence preserves ACTL$^{*J}_{-X}$ formulae. Using induction on the structure of φ it is easy to show that J-stuttering simulation equivalence also preserves ACTL*K$^J_{-X}$ formulae. □

It follows that if we are able to show that the cutoff instance $\mathcal{T}(c)$ is J-stuttering equivalent to an arbitrary system instance $\mathcal{T}(n)$, then we can use $\mathcal{T}(c)$ to check whether a formula $\bigwedge_J \varphi(J) \in$ IACTL*K$_{-X}$ holds for an arbitrary number of agents.

3. MODEL CHECKING PIIS

We now present a technique for model checking parameterised interleaved interpreted systems. In particular, we propose an efficient and automated methodology for answering the following verification query:

$$\forall n \geq |J| : \mathcal{T}(n) \models \psi, \text{ where } \psi = \bigwedge_J \varphi(J) \in \text{IACTL}^*\text{K}_{-X}$$

In other words we would like to check whether the property $\varphi(C)$ holds for *any number* $n \geq |J|$ of agents in the system, and for any $|J|$-tuple C of distinct agents. Note that the number of systems we would like to verify is unbounded. Therefore, traditional techniques to handle the state explosion problem cannot be used here. The key observation is that in certain circumstances it is sufficient to analyse only a finite number of systems to deduce properties about any larger system. Inspired by the work on cutoffs in the context of reactive systems [8, 10, 14], we say that a *MAS cutoff* c is a value of the system parameter for which the system instance $\mathcal{T}(c)$ exhibits all the behaviour admitted by any system instance $\mathcal{T}(n)$, $n \geq c$, with respect to a certain specification being considered.

Definition 3.1. *(MAS Cutoff) Let $\mathcal{T}(n)$ be a parameterised interleaved interpreted system and let $\psi \in IACTL^*K_{-X}$ be of the form $\bigwedge_J \varphi(J)$. A natural number $c \geq |J|$ is said to be a* MAS cutoff *for ψ if $\mathcal{T}(c) \models \psi \Leftrightarrow \forall n \geq c : \mathcal{T}(n) \models \psi$.*

It follows that if a cutoff can be identified, then model checking an infinite family of systems can be reduced to model checking all system instances up to the cutoff. Cutoff identification methodologies are typically accompanied by the following shortcomings: (i) either the cut-off is not guaranteed to be the smallest [8, 10], or (ii) the cut-off is not guaranteed to exist leading to incomplete methodologies [14]. By contrast, we here present a sound and complete methodology for identifying the *smallest cutoff* in model checking PIIS. As we will see later, to achieve this we pay a price in terms of the range of systems we can apply our results to. The following lemma shows that the smallest cutoff for an ACTL*K$^{\{1,\cdots,k\}}_{-X}$ formula φ is precisely k, the total number of agents appearing in the epistemic modalities and the propositions.

Lemma 3.2. *If $\varphi(\{1, \cdots, k\})$ is an $ACTL^*K^{\{1,\cdots,k\}}_{-X}$ formula, then $\mathcal{T}(n) \models \varphi(\{1, \cdots, k\})$ iff $\mathcal{T}(k) \models \varphi(\{1, \cdots, k\})$, for all $n \geq k$.*

Proof. Choose an arbitrary $n \geq k$. Let $[n] = \{1, \cdots, n\}$ and $[k+1, n] = [n] \setminus [k]$. We show that $\mathcal{T}(n) \leq_{[k]ss} \mathcal{T}(k)$ and $\mathcal{T}(k) \leq_{[k]ss} \mathcal{T}(n)$. The lemma then follows.

($\Rightarrow$ ($\mathcal{T}(n) \leq_{[k]ss} \mathcal{T}(k)$)) Define a relation $\sim_{[k]ss} = \{(g, g') \in G^n \times G^k \mid g_{[k]} = g'\}$. We show that $\sim_{[k]ss}$ is a $[k]$-stuttering simulation between $\mathcal{T}(n)$ and $\mathcal{T}(k)$. Let $g \sim_{[k]ss} g'$. Suppose

that $g_i = g_i^1$ for some $i \in [k]$ and let $g'^1 = g_{[k]}^1$. We have that $g'_i = g_i'^1$ and $g^1 \sim_{[k]ss} g'^1$. Now let $\pi = g^1 a^1 g^2 a^2 g^3 \cdots$ be a path of $\mathcal{T}(n)$ originating from $g^1 = g$. We construct a path ρ of $\mathcal{T}(k)$ originating from g' as required by $[k]$-stuttering simulation. Let $\rho = g_{[k]}^1 a'^1 g_{[k]}^2 a'^2 g_{[k]}^3 \cdots$, where $a'^j = a^j$ if $a^j \in \bigcup_{i\in[k]} Act_i$ and $a'^j = \epsilon$ otherwise, be the sequence obtained by the projection of π onto $[k]$. By assumption on the joint silent action, ρ is a valid path of $\mathcal{T}(k)$. We define a partition $B_1, B_2, \cdots$ of π and a partition $B'_1, B'_2, \cdots$ of ρ such that $|B_j| = |B'_j| = 1$ for each $j \geq 1$. It follows that $B_j \sim_{[k]ss} B'_j$ for each $j \geq 1$. Therefore, $\mathcal{T}(n) \leq_{[k]ss} \mathcal{T}(k)$.

($\Leftarrow (\mathcal{T}(k) \leq_{[k]ss} \mathcal{T}(n))$) The idea is to allow every agent $i \in [k+1, n]$ in $\mathcal{T}(n)$ to mimic agent 1 (in $\mathcal{T}(n)$). For this purpose, define a relation $\sim_{[k]ss}$ by

$$\Big\{(g, g') \in G^k \times G^n \mid g = g'_{[k]} \wedge \exists a^* \in Act^A : \forall i \in [k+1, n] : \big(a_i^* \in P_i(g'_i) \wedge tl(t_i(g'_i, a_i^*)) = tl(g'_1)\big) \vee tl(g'_i) = tl(g'_1)\Big\}$$

If $g \sim_{[k]ss} g'$, then each agent $i \in [k+1, n]$ in g' is either at the same local state with agent 1 in g' or agent i is able to change its state to the state of agent 1 by performing the asynchronous action a_i^*. We show that $\sim_{[k]ss}$ is a $[k]$-stuttering simulation between $\mathcal{T}(k)$ and $\mathcal{T}(n)$. Let $g \sim_{[k]ss} g'$. Simulation requirement 2(a) follows by a similar argument used in the left to right direction of the lemma. For simulation requirement 2(b), note that since the global evolution function is deterministic, a path $g^1 a^1 g^2 a^2 \cdots$ is uniquely defined by the sequence $g^1 a^1 a^2 \cdots$. We inductively define a function f which maps a path $\rho = g^1 a^1 g^2 a^2 g^3 \cdots$, $g^1 = g$, in $\mathcal{T}(k)$ into a path in $\mathcal{T}(n)$.

- $f(g^1 a^1 g^2 a^2 g^3 \cdots) = g' a_{j_1}^* \cdots a_{j_d}^* f(a^1 g^2 a^2 g^3)$, where $\{j_1 \cdots j_d\} = \{i \in [k+1, n] \mid tl(g'_i) \neq tl(g'_1)\}$;
- $f(a^1 g^2 a^2 g^3 \cdots) = a^1 f(a^2 g^3 \cdots)$, if $a^1 \notin Act_1^A$;
- $f(a^1 g^2 a^2 g^3 \cdots) = a^1 tl(a^1)_{k+1} \cdots tl(a^1)_n f(a^2 g^3 \cdots)$, if $a^1 \in Act_1^A$;

We partition π into singleton blocks $B_1, B_2, \cdots$ and we partition $f(\pi) = g^1 a^1 \cdots$ into the sequence $B'_1, B'_2, \cdots$, where $B'_i = g^j$, if $a^{j-1} \in \bigcup_{z\in[2,k]} Act_z$, and $B'_i = g^j \cdots g^{j+d}$, if $a^{j-1}, \cdots, a^{j+d-1} \in \bigcup_{z\in\{1\}\cup[k+1,n]} Act_z$ and $a^{j+d} \in \bigcup_{z\in[k]} Act_z$. It follows that $B_j \sim_{[k]ss} B'_j$, therefore, $\mathcal{T}(k) \leq_{[k]ss} \mathcal{T}(n)$. □

A consequence of the above lemma is the following:

THEOREM 3.3. *Let ψ be an IACTL*K$_{-X}$ formula of the form $\bigwedge_J \varphi(J)$. Then, $\forall n \geq |J| : \mathcal{T}(n) \models \psi$ iff $\mathcal{T}(|J|) \models \psi$.*

PROOF. By exploiting symmetry (Lemma 2.9), it suffices to prove the result for $\varphi([|J|])$ (Lemma 3.2). □

The above theorem is our main theoretical result. It follows that to verify a formula $\bigwedge_J \varphi(J)$ on all system instances, it suffices to verify the formula $\varphi([J])$ for the system instance $\mathcal{T}(|J|)$. Since in the MAS literature most properties are expressed by using one or two epistemic and propositional indices, this dramatically improves our verification abilities. Furthermore we can combine this technique with others available in the literature. Specifically, upon obtaining the instance $\mathcal{T}(|J|)$ we can further apply partial order reductions [16], abstraction [7], data symmetry reduction [6], etc., to further reduce the state space of the model.

COROLLARY 3.4. *Model checking parameterised interleaved interpreted systems against IACTL*K formulae of the form $\bigwedge_J \varphi(J)$ is decidable.*

PROOF. By Theorem 3.3, it suffices to model check the system instance of $|J|$ agents against $\varphi([|J|])$. □

The above is in line with literature in reactive systems [3, 8, 10, 9] where, although verification of parameterised systems is known to be undecidable in general [2], decidable fragments have been obtained by imposing restrictions on the systems and the properties studied.

4. EXAMPLE: AUTONOMOUS ROBOTS

For illustration purposes we exemplify the theory presented above on a parameterised variant of the autonomous robot example. We assume an arbitrary number of robots each running along its own track and each equipped with its own faulty sensor. The environment may move all non-halted robots one position forward at each time step. We represent this scenario by means of PIIS. We arbitrarily choose eight distinct locations; note that the number of locations does not affect the scenario as long as it is greater than four. Since we use interleaving semantics, we assume that the environment moves each robot in sequence; however, we insist on the environment to move all robots before moving a robot twice.

We proceed to define the template agent $\mathcal{T}$. A template state is a 4-tuple $l = (p, s, h, m)$, where p and s represent the position of the robot and the value of its sensor respectively, h represents whether or not the robot has halted, and m is a binary variable representing whether or not the environment has moved the robot in an interleaving sequence (a sequence in which the environment moves all non-halted robots from position q to $q+1$). Therefore, $L = \{(p, s, h, m) \mid 0 \leq p, s \leq 7 \text{ and } h, m \in \{\top, \bot\}\}$ is the set of template states from which we define $\iota = (0, 0, \bot, \bot)$ as the initial template state.

A robot can either do nothing or halt; the set Act^A of asynchronous template actions is $Act^A = \{null^=, null^+, null^-, halt\}$; the *null* actions represent the environment moving the robot a position forward and either providing a correct sensor reading ($null^=$) or not ($null^+, null^-$). A robot can move to position $q+1$ only if all non-halted robots have moved to position q; the unique synchronous template action *n_s* (next step) synchronises all robots before the environment can move a robot. As it will be clear below, when a *null* action is performed at position q, then m is set to $\top$ and the protocol selects the action *n_s* thereby disallowing a robot to move at position $q+1$ before all robots have moved to position q.

The template protocol P selects one of the *null* actions at position q when $m = \bot$ and the sensor reading is less than 3. The synchronous action *n_s* is the only allowed action when $m = \top$. Whenever the sensor reading is greater than 2, the halting condition is satisfied; therefore, the protocol selects the *halt* action: $P((p < 7, s < 3, h = \bot, m = \bot)) = \{null^=, null^+, null^-\}$; $P((p = *, s = *, h = *, m = \top)) = \{n\_s\}$; $P((p = *, s \geq 3, h = \bot, m = \bot)) = \{halt\}$, where $*$ expresses any value. The template evolution function contains the following transitions: $(p, s, \bot, \bot) \overset{null^=}{\rightarrow} (p+1, p+1, \bot, \top)$; $(p, s, \bot, \bot) \overset{null^+}{\rightarrow} (p+1, p+2, \bot, \top)$; $(p, s, \bot, \bot) \overset{null^-}{\rightarrow} (p+1, p, \bot, \top)$; $(p, s, \bot, \top) \overset{n\_s}{\rightarrow} (p, s, \bot, \bot)$ and $(p, s, \bot, \bot) \overset{halt}{\rightarrow} (p, s, \top, \top)$.

We introduce the following atomic propositions: $AP = \{h(\text{halted}), g(\text{goal region}), r(\text{restricted region})\}$. The interpretation of these propositions is given by the following valuation function: $V(h) = \{l \in L \mid l_3 = \top\}$, $V(g) = \{l \in L \mid 2 \leq l_1 \leq 4\}$, and $V(r) = \{l \in L \mid 0 \leq l_1 \leq 1\}$.

We verify that the halting condition is sound and complete in the parameterised variant by verifying the formulae $\varphi_{AR2} = \bigwedge_{\{i\}} AG(g_i \rightarrow AG(g_i))$ and $\varphi_{AR3} = \bigwedge_{\{i\}} AG(r_i \rightarrow \neg h_i)$. The specification φ_{AR2} expresses that "for every robot i, if i is within the goal region, then i never exits the goal region". The formula φ_{AR3} states that "for every robot i, if i is within the restricted region, then i has not halted". Note that the combined state space for the systems to be checked is unbounded. Observe also that model checking the above specifications is equivalent to model checking the formulae $AG(g_1 \rightarrow AG(g_1)) \wedge \cdots \wedge AG(g_n \rightarrow AG(g_n))$ and $AG(r_1 \rightarrow \neg h_1) \wedge \cdots \wedge AG(r_n \rightarrow \neg h_n)$, on each system instance $\mathcal{T}(n)$, $n \geq 1$. This is clearly not possible to check via standard model checking techniques. However, by using Lemmas 2.9 and 3.2 we can deduce that the MAS cutoff is equal to 1 and reduce the problem to checking the formulas $AG(g_1 \rightarrow AG(g_1))$ and $AG(r_1 \rightarrow \neg h_1)$ on the system instance $\mathcal{T}(1)$. This is a simple problem: we can easily check the specifications are verified, thereby deducing that the parameterised queries are also satisfied.

To proceed in our analysis further, we can also verify the formula $\varphi_{AR1} = \bigwedge_{\{i,j\}} AG(h_i \rightarrow K_i g_j)$. Also we could check that a robot knows that every other robot halted at the same time: $\varphi_{AR4} = \bigwedge_{\{i,j\}} AG(h_i \rightarrow K_i h_j)$. Similarly to what above, the formulae φ_{AR1} and φ_{AR4} can be reduced through Lemma 2.9 to $\varphi'_{AR1} = AG(h_1 \rightarrow K_1 g_2)$ and $\varphi'_{AR4} = AG(h_1 \rightarrow K_1 h_2)$, which can be verified on the system instance $\mathcal{T}(2)$ obtained by using a cutoff equal to 2 through Lemma 3.2. Also in this case we can check the result on the much smaller model and verify that the formula φ'_{AR1} holds while φ'_{AR4} does not (since the sensor readings may differ). So we infer that φ_{AR1} holds on the unbounded system while φ_{AR4} does not.

5. EVALUATION

Implementation.

We have implemented the presented methodology as an extension to the open-source model checker MCMAS [17]. The extended model checker, also open-sourced and named MCMAS-P, is available from [1]. ISPL, the input language of MCMAS, was suitably extended to allow for the definition of PIIS and to support the specification of indexed formulae. The description of a PIIS in this language (called PISPL) includes the declaration of a template agent. This declaration differs from agent declarations in ISPL by having sections of asynchronous and synchronous actions, and an initial state section. The specifications supported by MCMAS-P are expressed in indexed ACTLK$_{-X}$.

Given a PIIS and a formula to be verified, MCMAS-P determines the cutoff c for the system as in Theorem 3.3 by counting the number of unique indices used in the specification to be tested. A concrete system of c agents, each an indexed copy of the template agent, is then automatically constructed and represented symbolically. The specifications are automatically reduced to formulae in ACTLK$_{-X}$, as described in Lemma 2.9. The OBDD-based algorithms utilised by MCMAS are then used to verify the system against the reduced ACTLK$_{-X}$ formulae. The BDD encoding of the joint protocol is different from that of MCMAS to enforce the interleaving semantics used here.

Experimental Results.

In order to evaluate the methodology presented, we considered the parameterised autonomous robot scenario against the specification φ_{AR1}. We used a PC with an Intel Core i7 processor clocked at 2.20 GHz, with 6144 KiB cache, and running 64-bit Fedora 17, kernel 3.3.4. The results are reported in Table 1. The *Robots* and *States* columns respectively show the system instance (number of robots) and its state space; the *Instantiations* column shows the number of the possible instantiations of φ_{AR3}, each to be verified by the model checker; the *Time* and *Memory* columns show the CPU time and memory usage respectively. These results show that, as expected, the state space and the length of the formulae to be verified grow exponentially with the number of agents in the system. As a consequence of this, verification quickly becomes unfeasible under the time and memory constraints. This is exemplified for the system of 90 robots, where MCMAS did not finish the model construction within the timeout of one hour. In addition to this, of course, plain model checking cannot ever ensure the property holds on a system of arbitrary many agents. In comparison MCMAS-P constructed and verified a system of 2 robots in under 0.1 seconds thereby showing the property holds for an unbounded number of agents.

6. CONCLUSIONS AND RELATED WORK

In this paper we have developed a technique to verify that a temporal-epistemic property holds in a MAS irrespective of the number of agents present in the system. The problem is undecidable in general but we have defined a suitable semantics for which we gave a sound and complete procedure for determining a cutoff for a system. To do so we have defined a suitable parameterised logic and developed stuttering-equivalence simulation results for it on PIIS. We find the result encouraging as it opens the way for the verification of a large number of protocols previously verified only for individual instances containing a limited number of agents. Open systems with an unbounded number of homogeneous participants, e.g., including negotiations and auctions, seem particularly suitable for this analysis.

Related Work.

Existing literature on parameterised verification [3, 19, 20, 10, 8, 14] is limited to reactive systems and plain temporal logics. Moreover, mainstream methodologies [10, 8, 14] do not guarantee soundness, completeness and the identification of the smallest cutoff at the same time. In [14] a cutoff is identified by enumerating the system instances and finding the smallest instance able to simulate a "special" structure which includes the behaviour of every instance. Although the technique is widely applicable and independent of the communication topology, a cutoff is not guaranteed to exist. Results closer to those in this paper are the sound and complete techniques put forward in [10, 8]. Similarly to this contribution, [10, 8] present stuttering-simulation results between the cutoff model and every system instance thereby

Model		Instantiations	Time (s)	Memory (KiB)
Robots	States			
2	201	2	0	9010
30	$1.260\,92 \times 10^{27}$	870	18	44744
60	$3.594\,02 \times 10^{57}$	3540	868	63894032
90	TIMEOUT	8010	TIMEOUT	TIMEOUT

Table 1: MCMAS verification results for φ_{AR1}.

ensuring soundness and completeness. However, the results in [10] are applicable to ring topologies only and the technique in [8] does not identify the smallest cutoff.

In addition to cutoffs, abstraction techniques have of course been used in parameterised verification. In [19] concrete states are counter abstracted; an abstract state is a tuple of counters, one for each local state, denoting the number of system participants in the state. This process can be automated, but it is only applicable to a narrow class of systems and it is restricted to liveness properties. Environmental abstraction [3] extends counter abstraction by counting the number of participants that satisfy a given predicate and, although achieving wider applicability, the methodology has not, to our knowledge, been automated yet. In [20] a *network invariant* is identified which exhibits the behaviour of all system instances; if the invariant satisfies a property, then the property is satisfied by all system instances. A network invariant, however, is not guaranteed to exist, and, even when it does, its identification is not automated. In addition, none of these works tackle epistemic logic, nor MAS semantics, as we do here.

Future Work.

A current limitation of the PIIS formalism is that agents cannot evolve differently depending on the environment's action. This limits the application of the technique to particular systems such different network topologies. In future work we plan to alleviate this limitation as well as apply the methodology here presented to protocols of practical interests.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

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

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

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

[4] E.M. Clarke, O. Grumberg, and M.C. Browne. Reasoning about networks with many identical finite state processes. *Inf. and Comput.*, 81(1):13–31, 1989.

[5] G. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 1999.

[6] M. Cohen, M. Dam, A. Lomuscio, and H. Qu. A symmetry reduction technique for model checking temporal-epistemic logic. In *Proc. of IJCAI'09*, pages 721–726, 2009.

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

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

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

[10] E.A. Emerson and K.S. Namjoshi. Reasoning about rings. In *Proc. of POPL'95*, pages 85–94. ACM, 1995.

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

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

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

[14] Y. Hanna, D. Samuelson, S. Basu, and H. Rajan. Automating cut-off for multi-parameterized systems. In *Proc. of ICFEM'10*, pages 338–354. Springer, 2010.

[15] M. Kacprzak, W. Nabiałek, A. Niewiadomski, et al. Verics 2007-a model checker for knowledge and real-time. *Fundam. Inform.*, 85(1):313–328, 2008.

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

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

[18] W. Penczek, M. Szreter, R. Gerth, and R. Kuiper. Improving partial order reductions for universal branching time properties. *Fundam. Inform.*, 43(1):245–267, 2000.

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

[20] P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In *Automatic Verification Methods for Finite State Systems*, pages 68–80. Springer, 1990.