Parameterised Verification of Infinite State Multi-Agent Systems via Predicate Abstraction

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

Abstract

We define a class of parameterised infinite state multi-agent systems (MAS) that is unbounded in both the number of agents composing the system and the domain of the variables encoding the agents. We analyse their verification problem by combining and extending existing techniques in parameterised model checking with predicate abstraction procedures. The resulting methodology addresses both forms of unboundedness and provides a technique for verifying unbounded MAS defined on infinite-state variables. We illustrate the effectiveness of the technique on an infinite-domain variant of an unbounded version of the train-gate-controller.

Introduction

Over the past decade considerable progress has been made in the development of techniques to verify multi-agent systems (MAS) against agent-based specifications. These include SAT-based and BDD-based verification methods (Kacprzak, Lomuscio, and Penczek 2004; Raimondi and Lomuscio 2005). Current model checkers, such as Verics, MCK and MCMAS (Kacprzak et al. 2008; Gammie and van der Meyden 2004; Lomuscio, Qu, and Raimondi 2015), can efficiently verify large state-spaces.

The methods developed differ in many aspects, including the specifications supported and the input language used to represent the MAS to be analysed. However, they all analyse finite-state MAS that have two fundamental assumptions: firstly, the number of agents is finite and known at design time; secondly, the descriptions of the agents in the system use variables with finite domain. While both these assumptions ensure that the verification problem remains decidable (even PTIME-complete in several cases), their applicability is hampered in real-world applications.

For example, when analysing open MAS where agents join and leave the system at run time it may not be possible to know at design time how many agents the system will have at runtime. Similarly, in robotic swarms, the number of agents in a swarm is not known at design time. Recently, proposals to tackle the unbounded nature of agents have been put forward (Kouvaros and Lomuscio 2013; 2016). Decidable cases have been identified and cut-offs and counter-abstraction methods have been developed so that systems composed with an unbounded number of agents can be analysed. In these techniques the agents can be described by variables with finite domain only.

Irrespective of the choice of the programming language, MAS are typically programmed by using variables with infinite domains (integers, reals, etc.). This makes it difficult to ensure that a finite-state model accurately represents the system to be verified. To overcome this predicate-abstraction methods for infinite-state models have recently been put forward (Lomuscio and Michaliszyn 2015; 2016).

These two streams of work can independently deal with two sorts of unboundedness and potential undecidability of the verification problem: infinite state variables and unbounded number of agents. However, they work in isolation and cannot presently be combined. It follows that MAS with an unbounded number of components where each of them has at least one infinite-state variable cannot be analysed. In the present paper we develop a solution to this problem by introducing a technique that combines cut-off generation and predicate abstraction. The technique first uses predicate abstraction on the templates on the agents, thereby generating finite abstractions. Sufficient conditions are given for a cut-off of the system to be generated automatically. Having established a cut-off, the specification can then be checked on a three-valued semantics on all systems up to the cut-off. If the specification is found either to be true or false, then we can derive a conclusion on the original, infinite state system.

The rest of the paper is organised as follows. In Section 2, we give the syntax and semantics for the logic-based technique. In Section 3, we derive the main formal results and give an algorithm for the verification of MAS. We exemplify the methodology on a widely-adopted scenario in Section 4. We conclude in Section 5.

Related work. The technique here introduced is related to the two independent lines of work mentioned above (parameterised verification and predicate abstraction). None of them can address the infinite-state systems we work with in this paper. The technique here proposed can be seen as a combination of parameterised verification with predicate abstraction in the MAS domain. To achieve this, not only stronger conditions on cut-offs need to be identified, but also a three-valued semantics on these needs to be given so that they can be algorithmically checked.

Techniques to tackle both forms of unboundedness considered here were previously put forward in the context of reactive systems (John et al. 2012). However the semantics are incomparable to the semantics of this paper which are AI-based and thus require a different technical treatment. Also, (John et al. 2012) considers temporal specifications only and no attempt is made to analyse epistemic properties.

PIIS with Unbounded Variables

Parameterised interleaved interpreted systems (PIIS) extend interleaved interpreted systems (IIS) (Lomuscio, Penczek, and Qu 2010) to reason about the temporal-epistemic properties of asynchronous MAS with an unbounded number of agents (Kouvaros and Lomuscio 2013). Below we outline the PIIS semantics as presented in (Kouvaros and Lomuscio 2013), but, differently from the cited work, we here work on infinite state agents. There are, therefore, two forms of unboundedness in the systems we consider; one results from the domain of the variables encoding the agents; the other from the number of agents composing the system. We refer to these systems as PIIS with unbounded variables (PIIS^{UV}). A PIIS^{UV} consists of the descriptions of an *agent template* from which an unbounded number of homogeneous agents may be constructed and an *environment* in which the agents operate. Note the framework can accommodate a finite number of agent templates; for simplicity we do not pursue this here.

The agent template $T = \langle L, \iota, Act, P, t \rangle$ defines a non-empty, possibly infinite set of local states L, a unique initial state $\iota \in L$, and a non-empty, possibly infinite set of actions $Act = A \cup AE \cup GS$. Each action is either an *asynchronous action* (A) or an *agent-environment action* (AE) or a *global-synchronous action* (GS). Each type of action enables a different communication pattern between the concrete agents. In particular, asynchronous actions enable the asynchronous evolution of an agent; agent-environment actions enable pairwise synchronisation between one agent and the environment; global-synchronous actions enable full synchronisation among all the agents and the environment. The actions are performed in compliance with a protocol $P : L \rightarrow \mathcal{P}(Act)$ that selects which actions may be performed at a given state. The evolution of the local states is characterised by a transition function $t : L \times Act \rightarrow L$ returning the next local state given the current local state and the action performed at the state.

The environment $e = \langle L_e, \iota_e, Act_e, P_e, t_e \rangle$ is similarly associated with a non-empty, possibly infinite set of local states L_e, a unique initial state $\iota_e \in L_e$, a non-empty, possibly infinite set of actions $Act_e = AE \cup GS$, a protocol P_e, and a transition function t_e. Note that for the synchronisation purposes described above e admits the same agent-environment and global-synchronous actions with T.

We include the asynchronous "null' actions $null$ and $null_e$ to the sets of actions Act and Act_e respectively. It is assumed that: the protocol P is such that for every $l \in L$ we have that $null \in P(l)$ (i.e., the null action is enabled at every template state); the transition function t is such that $t(l, null) = l$ (i.e, the local state does not change whenever the null action is performed). The environment's null action $null_e$ is similarly described.

Definition 1 (PIIS^{UV})**.** *A parameterised interleaved interpreted system with unbounded variables is a tuple $\mathcal{S} = \langle T, e, \mathcal{V} \rangle$, where $\mathcal{V} : L \times AP \rightarrow \{\mathrm{tt}, \mathrm{ff}, \mathrm{uu}\}$ is a labelling function on the agent template's states for a set AP of atomic propositions.*

PIIS^{UV} give a generic description of an unbounded collection of concrete IIS, each one obtained by setting the parameter prescribing to the number of agents in the system. Given a PIIS^{UV} $\mathcal{S}$ and an integer $n \geq 1$, the IIS $\mathcal{S}(n)$ of n agents is the result of the composition of n copies of T with the environment. Atomic propositions in the concrete systems are interpreted over three truth values: true tt, false ff, and undefined uu. We say that a truth value x is defined whenever $x \neq \mathrm{uu}$. We write $\mathcal{A}$ for the set $\mathcal{A} = \{1, \dots, n\}$ of concrete agents instantiated from T. A *global state* $g = \langle l_1, \dots, l_n, l_e \rangle$ is a tuple of local states for all the agents in $\mathcal{S}(n)$; it describes the system at a particular instant of time. For a global state g we write $g.i$ to denote the local state of agent i in g. The system's global states evolve over time in compliance with the agents' local protocols and local evolution functions. The evolution is described by the global transition relation.

Definition 2 (Global transition relation)**.** *The global transition relation $R \subseteq G \times Act \times G$ on a set G of global states is defined as $(g, a, g') \in R$ iff one of the following holds:*

1. (Asynchronous transition). *(i) $a \in A$; (ii) there is $i \in \mathcal{A}$ with $a \in P(g.i)$ and $t(g.i, a) = g'.i$; (iii) for all $j \neq i$, $g.j = g'.j$.*
2. (Agent-environment transition). *(i) $a \in AE$; (ii) there is $i \in \mathcal{A}$ with $a \in P(g.i)$ and $t(g.i, a) = g'.i$; (iii) $a \in P_e(g.e)$ and $t_e(g.e, a) = g'.e$; (iv) for all $j \neq i$, $j \neq e$, $g.j = g'.j$.*
3. (Global-synchronous transition). *(i) $a \in GS$; (ii) for every $i \in \mathcal{A}$ we have that $a \in P(g.i)$ and $t(g.i, a) = g'.i$; (iii) $a \in P_e(g.e)$ and $t_e(g.e, a) = g'.e$.*

Above R defines only one action to be performed at each time step. If this is an asynchronous action, then exactly one agent participates in the global transition; if it is an agent-environment action, then exactly one agent and the environment participate in the global transition; if it is a global-synchronous action, then all the agents and the environment participate in the global transition. The agents not participating in a global transition are assumed to perform the null action at each time step. Since a global transition may always be taken by means of a null action, R is serial.

We now define the concrete systems generated from $\mathcal{S}$.

Definition 3 (Concrete semantics)**.** *Given a PIIS^{UV} $\mathcal{S}$ and $n \geq 1$, the IIS $\mathcal{S}(n)$ is a tuple $\mathcal{S}(n) = \langle G, g_0, R, V \rangle$, where $G \subseteq L^n \times L_e$ is the set of reachable global states via the global transition relation R from the initial global state $g_0 = (\iota, \dots, \iota, \iota_e)$, and $V : G \times (AP \times \mathcal{A}) \rightarrow \{\mathrm{tt}, \mathrm{ff}, \mathrm{uu}\}$ is the labelling function on the global states defined as $V(g, (p, i)) = x$ iff $\mathcal{V}(g.i, p) = x$, where $g \in G$, $p \in AP$, $i \in \mathcal{A}$, $x \in \{\mathrm{tt}, \mathrm{ff}, \mathrm{uu}\}$.*

A PIIS^{UV} generates different IIS depending on the parameter for the system. Each system is composed of a different number of agents. The propositional variables in an IIS are indexed by each of the concrete agents; (p, i) holds in a global state if the agent i is at a local state labelled with p by the template labelling function. This will enable us to construct specifications independently of the size of the concrete system on which they are evaluated.

A path π is a sequence $\pi = g^0 a^0 g^1 a^1 g^2 \ldots$ such that $(g^i, a^i, g^{i+1}) \in R$ for every $i \geq 0$. We write $\pi(i)$ for the i-th state in π and $\pi(i, Act)$ for the i-th action in π. The set of all paths originating from a state g is denoted by $\Pi(g)$. A global state g is said to be reachable from a global state g' if there is a path $\pi \in \Pi(g')$ with $\pi(i) = g$ for some $i \geq 0$.

Example 1 (Train-gate-controller)**.** *We exemplify the technical notions introduced above on a variant of the train-gate-controller (TGC) (Alur, Henzinger, and Kupferman 2002) where both the number of trains and some of the domains for the variables in the trains' programs are unbounded. The system is composed of a controller and an arbitrary number of trains. Each train runs along a circular track and all tracks pass through a narrow tunnel. The tunnel can accommodate only one train to be in it at any time. Both sides of the tunnel are equipped with traffic lights which can be either green or red. To enter the tunnel each train communicates to the controller a counter representing the number of times the train has already entered the tunnel. The controller operates the colour of the traffic lights instructing the trains enter and exit the tunnel; in doing so priority is given to the train that communicated the smallest counter. We are here not interested in strategic play and we assume truthful communication.*

We model the TGC as a PIIS^{UV} $\mathfrak{S} = \langle T, e, \mathcal{V} \rangle$, where T represents the trains and e represents the controller. These are defined as follows.

- $L = \{(away, i), (wait, i), (tunnel, i) : i \in \mathbb{N}\}$. *A train may be away from the tunnel, waiting to enter the tunnel, or in the tunnel; in all cases i represents the train's counter.* $L_e = \{(green, \mu), (red, \mu) : \mu \in \mathbb{N}^*\}$, *where μ is a sorted array representing the counters of the trains that have requested to enter the tunnel.*
- $\iota = (away, 0)$, $\iota_e = (green, \epsilon)$. *Initially the trains are away and there are no pending requests to enter the tunnel.*
- $Act = Act_e = \{(signal, i), (enter, i) : i \in \mathbb{N}\} \cup \{exit\}$. *A train can request to enter the tunnel by signalling its counter, enter the tunnel, and exit the tunnel. All actions are agent-environment actions.*
-
  - $P((away, i)) = \{(signal, i)\}$ *and* $t((away, i), (signal, i)) = (wait, i)$. *Whenever the train is away from the tunnel it can request to enter it and go into the waiting state.*
  - $P((wait, i)) = \{(enter, i)\}$ *and* $t((wait, i), (enter, i)) = (tunnel, i + 1)$. *If in the waiting state the train can enter the tunnel; in doing so it increases its counter by one.*
  - $P((tunnel, i)) = \{exit\}$ *and* $t((tunnel, i), exit) = (wait, i)$. *If in the tunnel, the train can exit the tunnel and go into the away state.*
  - $P_e((green, \mu)) = \{(signal, i) : i \in \mathbb{N}\} \cup \{(enter, \mu[0])\}$, $t_e((green, \mu), (signal, i)) = (green, insert(\mu, i))$, *and* $t_e((green, \mu), (enter, i)) = (red, remove(\mu, i))$. *The environment handles requests from all trains but it only accepts the one having the smallest counter.*
  - $P_e((red, \mu)) = \{exit\}$ *and* $t_e((red, \mu), exit) = (green, \mu)$. *Whenever a train is in the tunnel the controller does not handle any new requests.*

Specification language

We express specifications in two logics built on the same syntax but with different semantics: the two-valued logic indexed ACTLK$^{2v}_{-X}$ and the three-valued logic indexed ACTLK$^{3v}_{-X}$. The logic indexed ACTLK$_{-X}$ extends ACTLK$_{-X}$ by introducing indexed atomic propositions and indexed epistemic modalities; intuitively, indexed ACTLK$_{-X}$ formulae quantify over the concrete agents (Kouvaros and Lomuscio 2016). ACTLK$_{-X}$ is universal fragment of the temporal-epistemic logic CTLK without the next time operator. Given a set AP of atomic propositions, and a set VAR of variable symbols, indexed ACTLK$^{2v}_{-X}$ and indexed ACTLK$^{3v}_{-X}$ formulae are defined by the following BNF grammar:

$$\phi ::= (p, v) \mid \neg(p, v) \mid \phi \wedge \phi \mid \phi \vee \phi \mid A(\phi U \phi) \mid A(\phi R \phi) \mid K_v \phi \mid \forall v \phi$$

where $p \in AP$ and $v \in VAR$. The epistemic modality $K_v \phi$ is read as "agent v knows that ϕ" (Fagin et al. 1995). The temporal modality $A(\phi U \psi)$ stands for "for all paths, at some point ψ holds and before then ϕ is true along the path"; and $A(\phi R \psi)$ denotes "for all paths, ψ holds along the path up to and including the point when ϕ becomes true in the path".

A variable appearing in an indexed ACTLK$_{-X}$ formula is said to be free if it is not in the scope of a universal quantifier. An ACTLK$_{-X}$ formula is said to be a sentence if there are no free variables appearing in the formula. We here consider only ACTLK$_{-X}$ sentences. We say that an ACTLK$_{-X}$ sentence is an m-indexed formula if there are precisely m variables from VAR appearing in the formula.

The interpretation of the temporal modalities on an IIS is given by means of the global transition relation (Clarke, Grumberg, and Peled 1999), and the epistemic modalities are interpreted by using the epistemic possibility relations (Fagin et al. 1995). The epistemic possibility relation for an agent i is defined as follows: $\sim_i = \{(g, g') \in G \times G : g.i = g'.i\}$. In the following we assume the Kleene semantics for boolean connectives and report the three-valued satisfaction relation $\models^3$. The two valued satisfaction relation $\models^2$ can be derived from $\models^3$ by restricting to the clauses for tt and classical negation. We write $((\mathcal{S}(n), g) \models^3 \phi) = x$ to mean that the formula ϕ is evaluated to x at g. If $\mathcal{S}(n)$ is clear, then we simplify the notation to $(g \models^3 \phi) = x$.

Definition 4 (Satisfaction)**.** *The 3-valued satisfaction relation $\models^3$ for an IIS $\mathcal{S}(n)$, a global state g of $\mathcal{S}(n)$, and a formula ϕ is defined as follows:*

- $(g \models^3 (p,i)) = V(g,(p,i))$.
- $(g \models^3 \phi_1 \wedge \phi_2) = \mathrm{tt}$ *iff* $(g \models^3 \phi_1) = \mathrm{tt}$ *and* $(g \models^3 \phi_2) = \mathrm{tt}$*;*
- $(g \models^3 \phi_1 \wedge \phi_2) = \mathrm{ff}$ *iff either* $(g \models^3 \phi_1) = \mathrm{ff}$ *or* $(g \models^3 \phi_2) = \mathrm{ff}$.
- $(g \models^3 \phi_1 \vee \phi_2) = \mathrm{tt}$ *iff either* $(g \models^3 \phi_1) = \mathrm{tt}$ *or* $(g \models^3 \phi_2) = \mathrm{tt}$*;*
- $(g \models^3 \phi_1 \vee \phi_2) = \mathrm{ff}$ *iff* $(g \models^3 \phi_1) = \mathrm{ff}$ *and* $(g \models^3 \phi_2) = \mathrm{ff}$.
- $(g \models^3 A(\phi_1 U \phi_2)) = \mathrm{tt}$ *iff for all* $\pi \in \Pi(g)$*, there is* $i \geq 0$ *s.t.* $(\pi(i) \models^3 \phi_2) = \mathrm{tt}$ *and for all* $j < i$ $(\pi(j) \models^3 \phi_1) = \mathrm{tt}$.
- $(g \models^3 A(\phi_1 U \phi_2)) = \mathrm{ff}$ *iff there is* $\pi \in \Pi(g)$ *s.t. for all* $i \geq 0$ *we have* $(\pi(i) \models^3 \phi_2) = \mathrm{ff}$ *or there is* $j < i$ *with* $(\pi(j) \models^3 \phi_1) = \mathrm{ff}$.
- $(g \models^3 A(\phi_1 R \phi_2)) = \mathrm{tt}$ *iff for all* $\pi \in \Pi(g)$*, either there is some* $i \geq 0$ *with* $(\pi(i) \models^3 \phi_1) = \mathrm{tt}$ *and for all* $j \leq i$ $(\pi(j) \models^3 \phi_2) = \mathrm{tt}$*; or for all* $i \geq 0$ $(\pi(i) \models^3 \phi_2) = \mathrm{tt}$.
- $(g \models^3 A(\phi_1 R \phi_2)) = \mathrm{ff}$ *iff there is* $\pi \in \Pi(g)$ *s.t. there is* $i \geq 0$ *with* $(\pi(i) \models^3 \phi_2) = \mathrm{ff}$*, and for all* $i \geq 1$ *we have* $(p(i) \models^3 \phi_1) = \mathrm{ff}$ *or* $(p(j) \models^3 \phi_2) = \mathrm{ff}$ *for some* $j \leq i$*.*
- $(g \models^3 K_i \phi) = \mathrm{tt}$ *iff* $(g' \models^3 K_i \phi) = \mathrm{tt}$ *for all* g' *with* $g \sim_i g'$.
- $(g \models^3 k_i \phi) = \mathrm{ff}$ *iff* $(g \models^3 \phi) = \mathrm{ff}$.
- $(g \models^3 \forall v \phi) = \mathrm{tt}$ *iff* $(g \models^3 \phi[v \mapsto ag]) = \mathrm{tt}$ *for all* $ag \in \{1, \ldots, n\}$.
- $(g \models^3 \forall v \phi) = \mathrm{ff}$ *iff* $(g \models^3 \phi[v \mapsto ag]) = \mathrm{ff}$ *for some* $ag \in \{1, \ldots, n\}$.
- *In all other cases the value of a formula is undefined.*

Remark 1. *Following (Lomuscio and Michaliszyn 2015) the above defines an epistemic formula* $K_i \phi$ *as* ff *at a state if* ϕ *is* ff *at the state. While this is stronger than the standard definition (that assigns* $(g \models^3 K_i \phi) = \mathrm{ff}$ *iff there is* g' *with* $g' \sim_i g$ *and* $(g' \models^3 \phi) = \mathrm{ff}$*), it is crucial for preserving the value of a formula from the abstract models to the concrete ones (Theorem 2).*

ACTLK$_{-X}$ generalises indexed CTL (Clarke, Grumberg, and Browne 1989), a parametric variant of CTL that introduces quantification operators over the system components. In addition to the next-time operator, the unrestricted nesting of the quantification operators can be used to represent the actual number of participants in the system (Clarke, Grumberg, and Browne 1989), thereby making the verification problem undecidable (Clarke et al. 2004). To circumvent this, indexed CTL typically excludes the next-time operator and is restricted to its prenex fragment in which all the quantifiers appear at the front of the formula (Aminof et al. 2014). In light of this, for the rest of the paper, we consider $\overline{m}$-indexed ACTLK$_{-X}$ formulae complying to the following schema:

$$\forall v_1 \ldots \forall v_m \left(\bigwedge_{i,j \in \{1,\ldots,m\}} \neg(v_i = v_j) \rightarrow \phi(\{v_1, \ldots, v_m\}) \right)$$

where ϕ is an ACTLK$_{-X}$ formula with no quantifiers and without the next-time operator that is built from precisely the variables $v_1, \ldots, v_m$. We simply write ϕ to denote an $\overline{m}$-indexed formula of the above schema.

Example 2. *Consider the specification "whenever a train is in the tunnel it knows that no other train is in the tunnel at the same time" of the train-gate-controller. This can be expressed by the following* 2*-indexed formula:*

$$\phi_{\mathfrak{G}} = \forall_v \forall_u AG((tunnel, v) \rightarrow K_v \neg(tunnel, u)).$$

The evaluation of $\phi_{\mathfrak{G}}$ *on a concrete system is determined by evaluating the conjunction of all its ground instantiations under any assignment for the variables. For instance, when evaluated on a concrete system with two agents,* $\phi_{\mathfrak{G}}$ *denotes the formula* $AG((tunnel, 1) \rightarrow K_1 \neg(tunnel, 2)) \wedge AG((tunnel, 2) \rightarrow K_2 \neg(tunnel, 1))$*.*

An ACTLK$_{-X}$ formula ϕ is said to be true in $\mathcal{S}(n)$, denoted $\mathcal{S}(n) \models^2 \phi$, if $(\mathcal{S}(n), g_0) \models^2 \phi$. We define $(\mathcal{S}(n) \models^3 \phi) = \mathrm{tt}$ if $((\mathcal{S}(n), g_0) \models^3 \phi) = \mathrm{tt}$, $(\mathcal{S}(n) \models^3 \phi) = \mathrm{ff}$ if $((\mathcal{S}(n), g_0) \models^3 \phi) = \mathrm{ff}$, and $(\mathcal{S}(n) \models^3 \phi) = \mathrm{uu}$ otherwise. Defined truth values are preserved from the 3-valued satisfaction relation to the 2-valued one.

Theorem 1 (Relation between $\models^2$ and $\models^3$)**.** *Let* $\mathcal{S}(n)$ *be an IIS,* g *a global state of* $\mathcal{S}(n)$*, and* ϕ *an ACTLK*$_{-X}$ *formula. The following hold:*

1. $(g \models^3 \phi) = \mathrm{tt} \Longrightarrow g \models^2 \phi$.
2. $(g \models^3 \phi) = \mathrm{ff} \Longrightarrow g \not\models^2 \phi$.

In the following we exploit the above result to define a procedure to solve the verification problem for PIIS^{UV}.

Parameterised Verification

We now put forward a methodology to assess the correctness of a MAS formalised as a PIIS^{UV}. The decision problem, generally known as the *parameterised model checking problem*, is to check that a given PIIS^{UV} meets its specifications irrespective of the number of agents in the system.

Definition 5 (PMCP)**.** *Given a PIIS*^{UV} $\mathcal{S}$ *and an* m*-indexed formula* ϕ*, the parameterised model checking problem (PMCP) is the decision problem of determining whether the following holds:*

$$\mathcal{S}(n) \models^2 \phi \textit{ for every } n > m.$$

If this holds, then ϕ *is said to be satisfied by* $\mathcal{S}$*; this is denoted by* $\mathcal{S} \models^2 \phi$*.*

The PCMP is in general undecidable even for finite-state templates (Apt and Kozen 1986). Moreover, since every concrete system has a possibly infinite state space, the plain model checking problem on any concrete system is also undecidable. Thus we face two challenges to address the verification problem for PIIS^{UV}: we need to bound the number of variables encoding a concrete system and the number of systems that need to be checked. To do the former we abstract the agent template and the environment using predicates derived from the PIIS^{UV} and the specifications under consideration. We show that defined truth values are preserved from the systems generated from the abstract PIIS^{UV} to the systems generated from the original PIIS^{UV}. As a result, the PMCP is reduced to checking an unbounded number of finite-state systems. To address the unbounded nature of the number of agents to be considered, we identify a sufficient condition between the agent template and the environment for determining a natural number, the *cut-off*, expressing the number of systems that is sufficient to verify in order to solve the PMCP. We show how this condition can be checked on the abstract PIIS^{UV}. Consequently the PMCP is reduced to checking the systems generated from the abstract PIIS^{UV} up to the cut-off system. We first describe the predicate abstraction method and then show how a cut-off can be determined.

Predicate abstraction

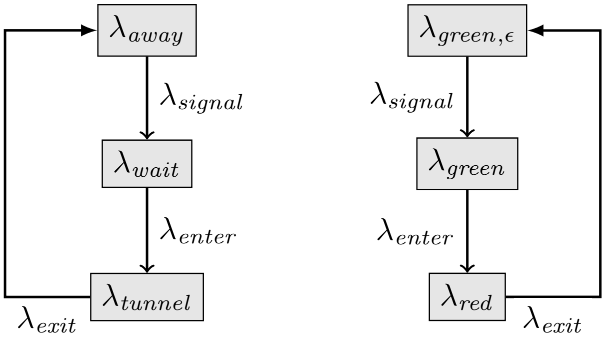
Assume an agent template T, a tuple $\overline{p_{s\_T}}$ of state predicates, a tuple $\overline{p_a}$ of asynchronous action predicates, a tuple $\overline{p_{ae}}$ of agent-environment action predicates, and a tuple $\overline{p_{gs}}$ of global-synchronous action predicates. Intuitively, each predicate represents a condition on the template's protocol or transition relation. The satisfaction of conjunctions λ of state predicates and their negation, called state cubes, on a template's state is denoted as $l \models \lambda$. Similarly, the satisfaction of conjunctions α of action predicates and their negation, called action cubes, on a template's action is denoted as $a \models \alpha$. A state (respectively, action) cube is satisfiable iff it is satisfied by some local state (respectively, action). The agent template is abstracted via predicates in the following way.

Definition 6 (Abstract agent template)**.** *Given an agent template T and a list of predicates $(\overline{p_{s\_T}}, \overline{p_a}, \overline{p_{ae}}, \overline{p_{gs}})$, the* abstract agent template *is the tuple $\hat{T} = \langle \hat{L}, \hat{\iota}, \hat{Act}, \hat{P}^{may}, \hat{P}^{must}, \hat{t}^{may}, \hat{t}^{must} \rangle$, where:*

- *$\hat{L}$ is the set of all satisfiable state cubes.*
- *$\hat{\iota}$ is the* initial state cube *satisfiable only by ι.*
- *$\hat{Act} = \hat{A} \cup \hat{AE} \cup \hat{GS}$ is the union of the sets of all satisfiable asynchronous, agent-environment and global-synchronous action cubes.*
- *the* may protocol $\hat{P}^{may}$ is defined as $\alpha \in \hat{P}^{may}(\lambda)$ iff there are $l \in L$, $a \in Act$ with $l \models \lambda$, $a \models \alpha$ and $a \in P(l)$.
- the *may transition relation* $\hat{t}^{may}$ is defined as $\hat{t}^{may}(\lambda, \alpha, \lambda')$ iff there are $l, l' \in L$, $a \in Act$ with $l \models \lambda$, $l' \models \lambda'$, $a \models \alpha$, and $t(l, a) = l'$.
- the *must protocol $\hat{P}^{must}$ is defined as $\alpha \in \hat{P}^{must}(\lambda)$ iff for every $l \in L$, $a \in Act$, if $l \models \lambda$ and $a \models \alpha$, then $a \in P(l)$.*
- *the* must transition relation *$\hat{t}^{must}$ is defined as $\hat{t}^{must}(\lambda, \alpha, \lambda')$ iff for all $l \in L$, $a \in Act$, if $l \models \lambda$ and $a \models \alpha$, then $t(l, a) = l'$ for some l' with $l' \models \lambda'$.*

Intuitively the *may* and *must* components of $\hat{T}$ are respectively over- and under-approximations of the temporal evolution of T. The abstract environment $\hat{e} = \langle \hat{L}_e, \hat{\iota}_e, \hat{Act}_e, \hat{P}_e^{may}, \hat{P}_e^{must}, \hat{t}_e^{may}, \hat{t}_e^{must} \rangle$ is similarly defined over $\overline{p\_ae}$, $\overline{p\_gs}$ and a tuple $\overline{p_{s\_e}}$ of state predicates.

Definition 7 (Abstract PIIS^{UV})**.** *Given a PIIS^{UV} $\mathcal{S}$ and a list of predicates $(\overline{p_{s\_T}}, \overline{p_a}, \overline{p_{ae}}, \overline{p_{gs}}, \overline{p_{s\_e}})$, the predicate abstraction of $\mathcal{S}$ is the tuple $\hat{\mathcal{S}} = \langle \hat{T}, \hat{e}, \hat{\mathcal{V}} \rangle$, where $\hat{T}$ is the abstract agent template w.r.t $\overline{p_{s\_T}}, \overline{p_{a\_ag}}, \overline{p_{ae}}, \overline{p_{gs}}$, $\hat{e}$ is the abstract environment w.r.t $\overline{p_{s\_e}}, \overline{p_{ae}}, \overline{p_{gs}}$, and $\hat{\mathcal{V}}$ is such that for any $\lambda \in \hat{L}$, $p \in AP$, $x \in \{\mathrm{tt}, \mathrm{ff}\}$, we have $\hat{\mathcal{V}}(\lambda, p) = x$ iff $\mathcal{V}(l, p) = x$ for all $l \in L$ with $l \models \lambda$.*



(a) Abstract train. (b) Abstract controller.

Figure 1: The abstract PIIS^{UV} of the TGC.

Example 3 (Train-gate-controller)**.** *Figure 1 depicts the abstract PIIS^{UV} $\hat{\mathfrak{G}} = \langle \hat{T}, \hat{e}, \hat{\mathcal{V}} \rangle$ of the train-gate-controller. $\hat{T}$ is constructed w.r.t the state predicates $\lambda_{away} = (away, 0)$, $\lambda_{wait} = (wait, 0)$, $\lambda_{tunnel} = (tunnel, 0)$, and the action predicates $\lambda_{signal} = (signal, 0)$, $\lambda_{enter} = (enter, 0)$, $\lambda_{exit} = exit$; $\hat{e}$ is built w.r.t the same action predicates and the state predicates $\lambda_{green,\epsilon} = (green, \epsilon)$, $\lambda_{green} = (green, 0)$, $\lambda_{red} = (red, \epsilon)$. In the figure the arrows represent both the may and must transition relations.*

Given a PIIS^{UV} $\mathcal{S}$ with predicate abstraction $\hat{\mathcal{S}}$ and an integer $n \geq 1$, the abstract IIS $\hat{\mathcal{S}}(n)$ of n abstract agents composes n copies of $\hat{T}$ with $\hat{e}$. Its construction is identical to the construction of the concrete systems, but it defines a *may global transition relation* $\hat{R}^{may}$ and a *must global transition relation* $\hat{R}^{must}$. $\hat{R}^{may}$ ($\hat{R}^{must}$, respectively) is defined as in Definition 2, but considering the may (must, respectively) protocols and transition functions. $\hat{R}^{may}$ is used to interpret the clauses for tt in Definition 4, whereas $\hat{R}^{must}$ is used to interpret the clauses for ff in Definition 4.

Definition 8 (Abstract semantics)**.** *Given a PIIS^{UV} $\mathcal{S}$ with predicate abstraction $\hat{\mathcal{S}}$ and $n \geq 1$, the* abstract IIS $\hat{\mathcal{S}}(n)$ *is a tuple $\hat{\mathcal{S}}(n) = \langle \hat{G}, \hat{g}_0, \hat{R}^{may}, \hat{R}^{must}, \hat{V} \rangle$, where $\hat{G}$, $\hat{g}_0$ and $\hat{V}$ are defined as in Definition 3.*

The abstract systems can be used to interpret ACTLK$_{-X}$ formulae as per the 3-valued semantics: for every $n \geq 1$ the evaluation of a formula to true (false, respectively) on $\hat{\mathcal{S}}(n)$ implies the evaluation of the formula to true (false, respectively) on $\mathcal{S}(n)$.

Theorem 2 (Preservation theorem)**.** *Let $\mathcal{S}$ be a PIIS^{UV} with predicate abstraction $\hat{\mathcal{S}}$, $n \geq 1$ an integer, and ϕ an ACTLK$_{-X}$ formula. The following hold:*

1. $(\hat{\mathcal{S}}(n) \models^3 \phi) = \mathrm{tt} \Longrightarrow (\mathcal{S}(n) \models^3 \phi) = \mathrm{tt}$*;*
2. $(\hat{\mathcal{S}}(n) \models^3 \phi) = \mathrm{ff} \Longrightarrow (\mathcal{S}(n) \models^3 \phi) = \mathrm{ff}$.

In view of the theorem above, methodologies can be devised to derive predicates automatically on the basis of the system and the specifications under examination. For instance, procedures for plain, non-parameterised, infinite state MAS were put forward in (Lomuscio and Michaliszyn 2016). Since the agent template and the environment can be viewed as agents in the typical MAS setting, the cited works can be adapted to automatically generate the abstract PIIS^{UV}. By doing so we would obtain an unbounded number of abstract, finite state interleaved interpreted systems. It follows that if the specification is $\mathtt{tt}$ in all abstractions, then the specification holds on the original PIIS^{UV}; if it is evaluated to $\mathtt{ff}$ in at least one abstract system, then the specification does not hold on the original PIIS^{UV}; otherwise, no conclusions can be drawn. Note the PMCP is still intractable since an unbounded number of abstract systems need to be checked. In the following we solve this problem by bounding the number of systems to be analysed.

Agent-environment simulation

We introduce a notion of simulation between the agent template and the environment. Intuitively, the states of the environment of a PIIS^{UV} admitting this simulation represent shared resources that can be accessed by the agents via agent-environment synchronisations. For a given PIIS^{UV} and an m-indexed formula we show that if this simulation exists, then to solve the PMCP it is sufficient to check the concrete system of m agents only. We first fix some notation.

Given a local state l we write $l \rightarrow l'$ to mean that there is an asynchronous action a and a state l' with $a \in P(l)$ and $t(l, a) = l'$. For a set of states X we use $X \rightarrow X'$ to denote that there is a state $l \in X$ such that $l \rightarrow l'$ and $X' = X \cup \{l'\}$. The reflexive and transitive closure of $\rightarrow$ is denoted by $\xrightarrow{*}$. Concretely, $X \xrightarrow{*} X'$ represents the set of local states X' in which an unbounded number of agents may asynchronously move into from X. Given a state l and either an agent-environment or a global-synchronous action a we write $l \xrightarrow{a} l'$ to mean that there is a state l' with $a \in P(l)$ and $t(l, a) = l'$. For a set of states X and an agent-environment action a we write $X \xrightarrow{a} X'$ to express that there is a state l in X with $l \xrightarrow{a} l'$ and $X' = X \setminus \{l\} \cup \{l'\}$. For a set of states X and a global-synchronous action a we use $X \xrightarrow{a} X'$ to mean that $X' = \left\{l' : \exists l \in X.\, l \xrightarrow{a} l'\right\}$. By $X \xrightarrow{*a} X'$ we mean that there is X'' with $X \xrightarrow{*} X'' \xrightarrow{a} X'$. In other words, $X \xrightarrow{*a} X'$ represents the set of local states X' that results from an unbounded sequence of asynchronous transitions from X followed by an agent-environment or a global-synchronous transition. When applied on the environment's state the above operators are interpreted in the same way using the environment's protocol and transition function. Finally, we use $\xrightarrow{?;\, x}$ ($\xrightarrow{!;\, x}$, respectively) to indicate that the operator $\xrightarrow{x}$ is applied to the abstract PIIS^{UV} using the may (must, respectively) protocol and transition function.

We now define an agent-environment simulation.

Definition 9 (Agent-environment simulation). *An* agent-environment *simulation between T and e is a relation $\mathfrak{R} \subseteq \mathcal{P}(L) \times L_e$ such that $(\{\iota\}, \iota_e) \in \mathfrak{R}$ and whenever $(X, l_e) \in \mathfrak{R}$ the following conditions hold:*

1. *If $X \xrightarrow{*a} X'$, then $l_e \xrightarrow{a} l'_e$ and $(X', l'_e) \in \mathfrak{R}$.*
2. *If $X \xrightarrow{*a^1} X^1 \xrightarrow{*a^2} \cdots \xrightarrow{*a^k} X^k$, $X \xrightarrow{*b} X'$ and $l_e \xrightarrow{a^1} l_e^1 \xrightarrow{a^2} \cdots \xrightarrow{a^k} l_e^k \xrightarrow{b} l_e^{k+1}$, then $(X \cup X^k, l_e^k) \in \mathfrak{R}$.*

We write $T \leq e$ to denote that there is an agent-environment simulation between T and e. In the rest of the paper we restrict our discussion to the subclass of PIIS^{UV} admitting an agent-environment simulation. Intuitively, this is the subclass of PIIS^{UV} in which global-synchronous actions determine a subclass of the shared resources the agents can access. We assume that global-synchronous actions are enabled at every state of the agent template. Upon performing a global-synchronous action the system updates the set of accessible shared resources. By condition 1 not only does the environment always allow this to happen, but also it always permits an agent to take the lock on a resource via agent-environment synchronisations. Following the lock on a resource the agent has to release the lock before another agent can synchronise with the environment. In line with this, whenever different agents may synchronise with the environment in successive time steps, the last synchronisation of the preceding agent is interpreted as resource-releasing; thus agent-environment synchrnonisations are not subsequently blocked by the environment, as expressed by condition 2.

This is a subclass of the PIIS studied in (Kouvaros and Lomuscio 2016). However, the methodology there presented considers finite state templates only, and it assumes that all agent-environment and global-synchronous actions are enabled at exactly one state of the environment. As a result, the original formulation of the agent-environment simulation cannot be applied to the present setting.

We now show that whenever $T \leq e$ the evaluation of an m-indexed formula on the system with m agents is equivalent to the evaluation of the formula on every bigger system. Integers following this property are commonly referred to as cut-offs (Emerson and Kahlon 2000).

Theorem 3 (Cut-off theorem). *Let $\mathcal{S} = \langle T, e, \mathcal{V}\rangle$ be a PIIS^{UV} with $T \leq e$, ϕ an m-indexed formula, and $n \geq m$ an integer. The following holds:*

$$\mathcal{S}(m) \models^2 \phi \textit{ iff } \mathcal{S}(n) \models^2 \phi.$$

By Theorem 3 and Theorem 2 the PMCP can be solved by checking the abstract system with m agents. This assumes that $T \leq e$ can be established. However T and e are possibly infinite state structures. To circumvent this, we give a three-valued semantics for the agent-environment simulation thereby enabling the simulation test to be performed on the abstract PIIS^{UV}. We write $(\hat{T} \leq \hat{e}) = \mathtt{tt}$ to mean that the abstract PIIS^{UV} admits an agent-environment simulation, $(\hat{T} \leq \hat{e}) = \mathtt{ff}$ to express that it does not, and $(\hat{T} \leq \hat{e}) = \mathtt{uu}$ to denote that it is unknown whether it does.

Definition 10 (Three-valued semantics for agent-environment simulations).

- $(\hat{T} \leq \hat{e}) = \mathtt{tt}$ *if there is a relation $\hat{\mathfrak{R}} \subseteq \mathcal{P}(\hat{L}) \times \hat{L}_e$ such that $(\{\hat{\iota}\}, \hat{\iota}_e) \in \hat{\mathfrak{R}}$ and whenever $(\hat{X}, \lambda_e) \in \hat{\mathfrak{R}}$ the following conditions hold:*

  1. *If* $\hat{X} \xrightarrow{?;*\alpha} \hat{X}'$*, then* $\lambda_e \xrightarrow{!;\alpha} \lambda'_e$ *and* $(\hat{X}', \lambda'_e) \in \hat{\mathfrak{R}}$.
  2. *If* $\hat{X} \xrightarrow{?;*\alpha^1} \hat{X}^1 \xrightarrow{?;*\alpha^2} \cdots \xrightarrow{?;*\alpha^k} \hat{X}^k$, $\hat{X} \xrightarrow{?;*b} \hat{X}'$ *and* $\lambda_e \xrightarrow{?;\alpha^1} \lambda_e^1 \xrightarrow{?;\alpha^2} \cdots \xrightarrow{?;\alpha^k} \lambda_e^k \xrightarrow{?;b} \lambda_e^{k+1}$*, then* $(\hat{X} \cup \hat{X}^k, \lambda_e^k) \in \hat{\mathfrak{R}}$.

- $(\hat{T} \leq \hat{e}) = \mathrm{ff}$ *if there is no relation* $\hat{\mathfrak{R}} \subseteq \mathcal{P}(\hat{L}) \times \hat{L}_e$ *such that* $(\{\hat{\iota}\}, \hat{\iota}_e) \in \hat{\mathfrak{R}}$ *and whenever* $(\hat{X}, \lambda_e) \in \hat{\mathfrak{R}}$ *the following conditions hold:*
  1. *If* $\hat{X} \xrightarrow{!;*\alpha} \hat{X}'$*, then* $\lambda_e \xrightarrow{?;\alpha} \lambda'_e$ *and* $(\hat{X}', \lambda'_e) \in \hat{\mathfrak{R}}$.
  2. *If* $\hat{X} \xrightarrow{!;*\alpha^1} \hat{X}^1 \xrightarrow{!;*\alpha^2} \cdots \xrightarrow{!;*\alpha^k} \hat{X}^k$*,* $\hat{X} \xrightarrow{!;*b} \hat{X}'$ *and* $\lambda_e \xrightarrow{!;\alpha^1} \lambda_e^1 \xrightarrow{!;\alpha^2} \cdots \xrightarrow{!;\alpha^k} \lambda_e^k \xrightarrow{!;b} \lambda_e^{k+1}$*, then* $(\hat{X} \cup \hat{X}^k, \lambda_e^k) \in \hat{\mathfrak{R}}$.
- $(\hat{T} \leq \hat{e}) = \mathrm{uu}$ *in all other cases.*

The abstract PIIS^{UV} can be used to perform the simulation test according to the three-valued semantics.

Theorem 4 (Simulation test)**.** *Let* $\mathcal{S}$ *be a PIIS*^{UV} *with predicate abstraction* $\hat{\mathcal{S}}$*. The following hold:*

1. $(\hat{T} \leq \hat{e}) = \mathrm{tt} \Longrightarrow T \leq e$.

2. $(\hat{T} \leq \hat{e}) = \mathrm{ff} \Longrightarrow T \not\leq e$.

With a successful simulation test the analysis of the abstract system with m agents is sufficient to establish the correctness of the original PIIS^{UV}.

Theorem 5 (PMC theorem)**.** *Let* $\mathcal{S} = \langle T, e, \mathcal{V} \rangle$ *be a PIIS*^{UV} *with predicate abstraction* $\hat{\mathcal{S}} = \langle \hat{T}, \hat{e}, \hat{\mathcal{V}} \rangle$ *such that* $(\hat{T} \leq \hat{e}) = \mathrm{tt}$*. Let* ϕ *be an* m*-indexed formula. The following hold:*

1. $(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{tt} \Longrightarrow \mathcal{S} \models^2 \phi$.

2. $(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{ff} \Longrightarrow \mathcal{S} \not\models^2 \phi$.

The above is the main result of the paper. It provides the underpinnings for a constructive, sound but incomplete methodology to solve the PMCP for infinite state MAS. Concretely, verification of PIIS^{UV} can be conducted as follows. Firstly, the abstract PIIS^{UV} is built from predicates derived from the original PIIS^{UV} and an m-indexed specification. Then it is checked whether or not the abstract PIIS^{UV} admits an agent-environment simulation. If so, then the abstract system with m agents is checked against the specification; otherwise, no conclusions can be drawn. If the value of the specification is defined on said abstract system, then we can deduce whether or not the specification is satisfied on the original PIIS^{UV}; if the value is undefined, then the satisfaction of the specification on the original PIIS^{UV} can not be determined. Note that there is no difficulty in performing refinement if either the specification or the agent-environment simulation is undefined by following the procedure in (Belardinelli, Lomuscio, and Michaliszyn 2016).

Example 4 (Train-gate-controller)**.** *Consider the relation* $\hat{\mathfrak{R}} = \{(\lambda_{away}, \lambda_{green,\epsilon}), (\lambda_{wait}, \lambda_{green}), (\lambda_{tunnel}, \lambda_{red})\}$ *between the abstract train* $\hat{T}$ *and the abstract controller* $\hat{e}$*.* $\hat{\mathfrak{R}}$ *satisfies all the conditions of the first clause of Definition 10. Therefore* $(\hat{T} \leq \hat{e}) = \mathrm{tt}$*. Hence, by Theorem 5, the abstract system of* 2 *trains can be used to establish the correctness of the train-gate-controller against* $\phi_{\mathfrak{G}}$*; this can be put into any epistemic model checker which would return* true.

Conclusions

In this paper we have introduced a methodology for verifying infinite-state MAS with an unbounded number of components. We have given a three-valued predicate abstraction methodology for deriving finite descriptions of the agents and established sufficient conditions for the derivations of cut-offs.

In future work we intend to implement the methodology here described.

Acknowledgments

The research described in this paper was supported by the EPSRC under grant EP/I00520X/1 and a Doctoral Prize Fellowship.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

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

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

Belardinelli, F.; Lomuscio, A.; and Michaliszyn, J. 2016. Agent-based refinement for predicate abstraction of multiagent systems. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI16).*

Clarke, E.; Talupur, M.; Touili, T.; and Veith, H. 2004. Verification by network decomposition. In *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR04)*, volume 3170 of *Lecture Notes in Computer Science*. Springer. 276–291.

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

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. Cambridge, Massachusetts: The MIT Press.

Emerson, E., and Kahlon, V. 2000. Reducing model checking of the many to the few. In *Proceedings of the 17th International Conference on Automated Deduction (CADE00)*, volume 1831 of *Lecture Notes in Computer Science*, 236–254. Springer.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about Knowledge*. Cambridge: MIT Press.

Gammie, P., and van der Meyden, R. 2004. MCK: Model checking the logic of knowledge. In *Proceedings of 16th*

International Conference on Computer Aided Verification (CAV04), volume 3114 of *Lecture Notes in Computer Science*, 479–483. Springer.

John, A.; Konnov, I.; U.Schmid; Veith, H.; and Widder, J. 2012. Counter attack on byzantine generals: Parameterized model checking of fault-tolerant distributed algorithms. *arXiv preprint arXiv:1210.3846*.

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

Kacprzak, M.; Lomuscio, A.; and Penczek, W. 2004. Verification of multiagent systems via unbounded model checking. In *Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS04)*, 638–645. ACM.

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

Kouvaros, P., and Lomuscio, A. 2016. Parameterised verification for multi-agent systems. *Artificial Intelligence* 234:152–189.

Lomuscio, A., and Michaliszyn, J. 2015. Verifying multi-agent systems by model checking three-valued abstractions. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS15)*, 189–198.

Lomuscio, A., and Michaliszyn, J. 2016. Verification of multi-agent systems via predicate abstraction against ATLK specifications. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS16)*.

Lomuscio, A.; Penczek, W.; and Qu, H. 2010. Partial order reduction for model checking interleaved multi-agent systems. *Fundamenta Informaticae* 101(1–2):71–90.

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

Raimondi, F., and Lomuscio, A. 2005. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic* 5(2):235–251.

Appendix

Theorem 1 (Relation between $\models^3$ and $\models^2$)**.** *Let $\mathcal{S}(n)$ be an IIS, g a global state of $\mathcal{S}(n)$, and ϕ an ACTLK$_{-X}$ formula. The following hold:*

1. $(g \models^3 \phi) = \mathrm{tt} \Longrightarrow g \models^2 \phi$*.*

2. $(g \models^3 \phi) = \mathrm{ff} \Longrightarrow g \not\models^2 \phi$*.*

Proof. The proof is by structural induction on ϕ.

1. - Suppose that $\phi = (p, i)$ and $(g \models^3 \phi) = \mathrm{tt}$. Then $V(g, (p, i)) = \mathrm{tt}$. Therefore $g \models^2 \phi$.
   - The boolean cases are straightforward and are therefore omitted from the proof.
   - Assume $\phi = A(\phi_1 U \phi_2)$ and $(g \models^3 \phi) = \mathrm{tt}$. Then for all $\pi \in \Pi(g)$ there is $i \geq 0$ with $(\pi(i) \models^3 \phi_2) = \mathrm{tt}$, and $(\pi(j) \models^3 \phi_1) = \mathrm{tt}$ for all $0 \leq j < i$. Let $\pi \in \Pi(g)$ be arbitrary. By the inductive hypothesis there is $i \geq 0$ with $\pi(i) \models^2 \phi_2$, and $\pi(j) \models^2 \phi_1$ for all $0 \leq j < i$. Consequently $g \models^2 \phi$.
   - The case of $\phi = A(\phi_1 R \phi_2)$ can be shown similarly to the case of $\phi = A(\phi_1 U \phi_2)$.
   - Consider $\phi = K_i \phi_1$ and $(g \models^3 \phi) = \mathrm{tt}$. Then $g' \sim_i g$ implies $(g' \models^3 \phi_1) = \mathrm{tt}$. Let $g' \in G$ with $g' \sim_i g$ be arbitrary. By the inductive hypothesis $g' \models^2 \phi_1$. Hence $g \models^2 \phi$.
2. - Suppose that $\phi = (p, i)$ and $(g \models^3 \phi) = \mathrm{ff}$. Then $V(g, (p, i)) = \mathrm{ff}$. Therefore $g \not\models^2 \phi$.
   - The boolean cases are straightforward and are therefore omitted from the proof.
   - Assume $\phi = A(\phi_1 U \phi_2)$ and $(g \models^3 \phi) = \mathrm{ff}$. Then there is $\pi \in \Pi(g)$ s.t. for all $i \geq 0$ we have $(\pi(i) \models^3 \phi_2) = \mathrm{ff}$ or there is $j < i$ with $(\pi(j) \models^3 \phi_1) = \mathrm{ff}$. By the inductive hypothesis for all $i \geq 0$ we have $\pi(i) \not\models^2 \phi_2$ or there is $j < i$ with $\pi(j) \not\models^2 \phi_1$. Consequently $g \not\models^2 \phi$.
   - The case of $\phi = A(\phi_1 R \phi_2)$ can be shown similarly to the case of $\phi = A(\phi_1 U \phi_2)$.
   - Consider $\phi = K_i \phi_1$ and $(g \models^3 \phi) = \mathrm{ff}$. Then $(g \models^3 \phi_1) = \mathrm{ff}$. By the inductive hypothesis $g \not\models^2 \phi_1$. Hence $g \not\models^2 \phi$.

□

Theorem 2 (Preservation theorem)**.** *Let $\mathcal{S}$ be a PIIS^{UV} with predicate abstraction $\hat{\mathcal{S}}$, $n \geq 1$ an integer, and ϕ an ACTLK$_{-X}$ formula. The following hold:*

1. $(\hat{\mathcal{S}}(n) \models^3 \phi) = \mathrm{tt} \Longrightarrow (\mathcal{S}(n) \models^3 \phi) = \mathrm{tt}$*.*

2. $(\hat{\mathcal{S}}(n) \models^3 \phi) = \mathrm{ff} \Longrightarrow (\mathcal{S}(n) \models^3 \phi) = \mathrm{ff}$*.*

Proof. Define the satisfaction of a tuple $\hat{g} = (\lambda_1, \ldots, \lambda_n)$ of state cubes on a global state g as $\forall i.\, g.i \models \hat{g}.i$. Denote this by $g \models \hat{g}$. We show by structural induction on ϕ that if $g \models \hat{g}$, then

$$(\hat{g} \models^3 \phi) = \mathrm{tt} \Rightarrow (g \models^3 \phi) = \mathrm{tt} \qquad \text{(a)}$$

and

$$(\hat{g} \models^3 \phi) = \mathrm{ff} \Rightarrow (g \models^3 \phi) = \mathrm{ff}. \qquad \text{(b)}$$

(a) - Suppose that $\phi = (p, i)$ and $(\hat{g} \models^3 \phi) = \mathrm{tt}$. Then for all g' with $g'.i \models \hat{g}.i$ we have that $(g' \models^3 \phi) = \mathrm{tt}$. Therefore $(g \models^3 \phi) = \mathrm{tt}$.
   - The boolean cases are straightforward and are therefore omitted from the proof.
   - Assume $\phi = A(\phi_1 U \phi_2)$ and $(\hat{g} \models^3 \phi) = \mathrm{tt}$. Then for all $\hat{\pi} \in \Pi^{may}(\hat{g})$ there is $i \geq 0$ with $(\hat{\pi}(i) \models^3 \phi_2) = \mathrm{tt}$, and $(\hat{\pi}(j) \models^3 \phi_1) = \mathrm{tt}$ for all $0 \leq j < i$. Let $\pi \in \Pi(g)$ be arbitrary. Define $\hat{\pi} = \lambda_1 \alpha_1 \ldots$ where, for each $i \geq 0$, $\pi(i) \models \lambda_i$ and $\pi(i, Act) \models$

α_i. By the definitions of $\hat{P}^{may}$ and $\hat{t}^{may}$, for each $i \geq 0$, $(\hat{\pi}(i), \hat{\pi}(i, Act), \hat{\pi}(i+1)) \in \hat{R}^{may}$, therefore $\hat{\pi} \in \hat{\Pi}^{may}(\hat{g})$. By the inductive hypothesis there is $i \geq 0$ with $(\pi(i) \models^3 \phi_2) = \text{tt}$, and $(\pi(j) \models^3 \phi_1) = \text{tt}$ for all $0 \leq j < i$. Consequently, as π was arbitrary, $(g \models^3 \phi) = \text{tt}$.

- The case of $\phi = A(\phi_1 R \phi_2)$ can be shown similarly to the case of $\phi = A(\phi_1 U \phi_2)$.
- Consider $\phi = K_i \phi_1$ and $(\hat{g} \models^3 \phi) = \text{tt}$. Then $\hat{g}' \sim_i \hat{g}$ implies $(\hat{g}' \models^3 \phi_1) = \text{tt}$. Let g' with $g' \sim_i g$ be arbitrary. By definition there is $\hat{g}'$ with $\hat{g}' \models g'$ and $\hat{g}' \sim_i \hat{g}$. By the inductive hypothesis $g' \models \phi_1$. Hence $(g \models^3 \phi) = \text{tt}$.

(b)
- Suppose that $\phi = (p, i)$ and $(\hat{g} \models^3 \phi) = \text{ff}$. Then all g' with $g'.i \models \hat{g}.i$ are such that $(g' \models^3 \phi) = \text{ff}$. Therefore $(g \models^3 \phi) = \text{ff}$.
- The boolean cases are straightforward and are therefore omitted from the proof.
- Assume $\phi = A(\phi_1 U \phi_2)$ and $(\hat{g} \models^3 \phi) = \text{ff}$. Then there is $\hat{\pi} \in \hat{\Pi}^{must}(\hat{g})$ s.t. for all $i \geq 0$ we have $(\hat{\pi}(i) \models^3 \phi_2) = \text{ff}$ or there is $j < i$ with $(\hat{\pi}(j) \models^3 \phi_1) = \text{ff}$. By the definitions of $\hat{P}^{must}$ and $\hat{t}^{must}$ there is $\pi \in \Pi(g)$ with $\pi(i) \models \hat{\pi}(i)$ and $\pi(i, Act) \models \hat{\pi}(i, Act)$ for each $i \geq 0$. By the inductive hypothesis for all $i \geq 0$ we have $(\pi(i) \models^3 \phi_2) = \text{ff}$ or there is $j < i$ with $(\pi(j) \models^3 \phi_1) = \text{ff}$. Consequently $(g \models^3 \phi) = \text{ff}$.
- The case of $\phi = A(\phi_1 R \phi_2)$ can be shown similarly to the case of $\phi = A(\phi_1 U \phi_2)$.
- Consider $\phi = K_i \phi_1$ and $(\hat{g} \models^3 \phi) = \text{ff}$. Then $(\hat{g} \models^3 \phi_1) = \text{ff}$. By the inductive hypothesis $(g \models^3 \phi_1) = \text{ff}$. Hence $(g \models^3 \phi) = \text{ff}$.

Consequently, since $g_0 \models \hat{g_0}$, (a) implies

1. $(\hat{\mathcal{S}}(n) \models^3 \phi) = \text{tt} \Longrightarrow (\mathcal{S}(n) \models^3 \phi) = \text{tt}$

and (b) implies

2. $(\hat{\mathcal{S}}(n) \models^3 \phi) = \text{ff} \Longrightarrow (\mathcal{S}(n) \models^3 \phi) = \text{ff}$.

□

Theorem 3 (Cutoff theorem). *Let $\mathcal{S} = \langle T, e, \mathcal{V} \rangle$ be a PIIS^{UV} with $T \leq e$, ϕ an m-indexed formula, and $n \geq m$ an integer. The following holds:*

$$\mathcal{S}(m) \models^2 \phi \textit{ iff } \mathcal{S}(n) \models^2 \phi.$$

To prove the cutoff theorem we require a notion of behavioural equivalence between system instances. Recall that m-indexed ACTLK$_{-X}$ formulae are preserved under m-stuttering-simulation (Kouvaros and Lomuscio 2016). Let $\mathcal{S}(n) = \langle G(n), g_0(n), R(n), V(n) \rangle$ and $\mathcal{S}(n') = \langle G(n'), g_0(n'), R(n'), V(n') \rangle$. $\mathcal{S}(n)$ is said to m-stuttering-simulate $\mathcal{S}(n')$ if there is a relation $\mathfrak{R} \subseteq G(n') \times G(n)$ such that $(g_0(n'), g_0(n)) \in \mathfrak{R}$ and whenever $(g', g) \in \mathfrak{R}$, then:

1. $V(n')(g', (p, i)) = \text{tt}$ iff $V(n)(g, (p, i)) = \text{tt}$ for all $p \in AP$, $i \in \{1, \ldots, m\}$;
2. for every $\pi' \in \Pi(g')$, there is a $\pi \in \Pi(g)$, a partition $B_1', B_2', \ldots$ of π', and a partition $B_1, B_2, \ldots$ of π such that for each $i \geq 1$, B_i' and B_i are nonempty and finite, and every state in B_i' is related by $\mathfrak{R}$ to every state in B_i.

If $\mathcal{S}(n)$ m-stuttering-simulates $\mathcal{S}(n')$, then $\mathcal{S}(n) \models^2 \phi$ implies that $\mathcal{S}(n') \models^2 \phi$ for any m-indexed formula ϕ (Kouvaros and Lomuscio 2016).

Proof. For the left to right direction we show that $\mathcal{S}(m)$ m-stuttering-simulates $\mathcal{S}(n)$. For the right to left direction we show that $\mathcal{S}(n)$ m-stuttering-simulates $\mathcal{S}(m)$.

- [$\Longrightarrow$] Define $\mathfrak{R} \subseteq G(n) \times G(m)$ as $(g, g') \in \mathfrak{R}$ if $g.i = g'.i$ for $i \in \{1, \ldots, m\}$. It is easy to see that $(g_0(n), g_0(m)) \in \mathfrak{R}$. Let $(g, g') \in \mathfrak{R}$ and $\pi \in \Pi(g)$ be arbitrary. We construct a path $\pi' \in \Pi(g')$ as required by m-stuttering simulation. Let $\pi' = g'$, $B_1 = g$ and $B_1' = g'$. Then for each $j \geq 1$ extent π' and define the next blocks as follows:
  - if $\pi(j, Act)$ is either an asynchronous action or an agent-environment action performed by an agent i in $\{1, \ldots, m\}$, then $\pi' = \pi(j, Act) g'^1$, where g'^1 is the state resulting from the agent i performing $\pi(j, Act)$ at g'. $B_{j+1} = \pi(j+1)$ and $B_{j+1}' = g'^1$.
  - if $\pi(j, Act)$ is either an asynchronous action or an agent-environment action performed by an agent in $\{m+1, \ldots, n\}$, then $\pi' = null g'$, where the asynchronous action $null$ can be performed by any agent. $B_{j+1} = \pi(j+1)$ and $B_{j+1}' = g'^1$.
  - if $\pi(j, Act)$ is a global-synchronous action, then $\pi' = \pi' \pi(j, Act) g'^1$, where g'^1 is the state resulting from all the agents performing $\pi(j, Act)$ in g'. $B_{j+1} = \pi(j+1)$ and $B_{j+1}' = g'^1$.

  By $T \leq e\, \pi'$ is a valid path. Also the state in each B_j is related by $\mathfrak{R}$ to the state in B_j'. Thus π' is as required by m-stuttering simulation. It follows that $\mathcal{S}(m)$ m-stuttering simulates $\mathcal{S}(n)$.
- [$\Longleftarrow$] Define $\mathfrak{R} \subseteq G(m) \times G(n)$ as $(g, g') \in \mathfrak{R}$ if $g.i = g'.i$ for $i \in \{1, \ldots, m\}$. It is easy to see that $(g_0(n), g_0(m)) \in \mathfrak{R}$. Let $(g, g') \in \mathfrak{R}$ and $\pi \in \Pi(g)$ be arbitrary. We construct a path $\pi' \in \Pi(g')$ as required by m-stuttering simulation. Let $\pi' = g'$, $B_1 = g$ and $B_1' = g'$. Then for each $j \geq 1$ extent π' and define the next blocks as follows:
  - if $\pi(j, Act)$ is either an asynchronous or agent-environment action performed by the agent i, then $\pi' = \pi' \pi(j, Act) g'^1$, where g'^1 is the state resulting from the agent i performing $\pi(j, Act)$ at g'. $B_{j+1} = \pi(j+1)$ and $B_{j+1}' = g'^1$.
  - if $\pi(j, Act)$ is a global-syncrhonous action, then $\pi' = \pi' \pi(j, Act) g'^1$, where g'^1 is the state resulting from all the agents performing $\pi(j, Act)$ at g'. $B_{j+1} = \pi(j+1)$ and $B_{j+1}' = g'^1$.

  It is easy to see that for each B_j, B_j', the state in B_j is related by $\mathfrak{R}$ to every state in B_j'. Thus π' is as required by m-stuttering simulation. It follows that $\mathcal{S}(n)$ m-stuttering simulates $\mathcal{S}(m)$.

□

Theorem 4 (Simulation test)**.** *Let $\mathcal{S}$ be a PIIS^{UV} with predicate abstraction $\hat{\mathcal{S}}$. The following hold:*

1. $(\hat{T} \leq \hat{e}) = \mathrm{tt} \Longrightarrow T \leq e$.

2. $(\hat{T} \leq \hat{e}) = \mathrm{ff} \Longrightarrow T \not\leq e$.

Proof.

1. Suppose that $(\hat{T} \leq \hat{e}) = \mathrm{tt}$. Then there is a relation $\hat{\mathfrak{R}} \subseteq \mathcal{P}(\hat{L}) \times \hat{L}_e$ such that $(\{\hat{\iota}\}, \hat{\iota}_e) \in \mathfrak{R}$ and whenever $\hat{X} \sim \lambda_e$ the following conditions hold:

   (a) If $\hat{X} \xrightarrow{?;*\alpha} \hat{X}'$, then $\lambda_e \xrightarrow{!;\alpha} \lambda'_e$ and $(\hat{X}', \lambda'_e) \in \hat{\mathfrak{R}}$.

   (b) If $\hat{X} \xrightarrow{?;*\alpha^1} \hat{X}^1 \xrightarrow{?;*\alpha^2} \cdots \xrightarrow{?;*\alpha^k} \hat{X}^k$, $\hat{X} \xrightarrow{?;*b} \hat{X}'$ and $\lambda_e \xrightarrow{?;\alpha^1} \lambda_e^1 \xrightarrow{?;\alpha^2} \cdots \xrightarrow{?;\alpha^k} \lambda_e^k \xrightarrow{?;b} \lambda_e^{k+1}$, then $(\hat{X} \cup \hat{X}^k, \lambda_e^k) \in \hat{\mathfrak{R}}$.

   We have to show that $T \leq e$. Define $\mathfrak{R} \subseteq \mathcal{P}(L) \times L_e$ as $(X, l_e) \in \mathfrak{R}$ iff there is a set $\hat{X}$ of state predicates and a state predicate λ_e such that:

   - $(\hat{X}, \lambda_e) \in \hat{\mathfrak{R}}$.
   - $l_e \models \lambda_e$.
   - for each $l \in X$ there is $\lambda \in \hat{X}$ with $l \models \lambda$, and for each $\lambda \in \hat{X}$ there is $l \in X$ with $l \models \lambda$. We write $X \equiv \hat{X}$ to denote this.

   We prove that $\mathfrak{R}$ is an agent-environment simulation between T and e. Since $(\{\hat{\iota}\}, \hat{\iota}_e) \in \hat{\mathfrak{R}}$ we have that $(\{\iota\}, \iota_e) \in \mathfrak{R}$. Let $(X, l_e) \in \mathfrak{R}$ by means of $\hat{X}$, λ_e be arbitrary.

   - Assume $X \xrightarrow{*a} X'$. We have to show that $l_e \xrightarrow{a} l'_e$ and $(X', l'_e) \in \mathfrak{R}$. By the definitions of $\hat{P}^{may}$ and $\hat{t}^{may}$ we have that $\hat{X} \xrightarrow{?;*\alpha} \hat{X}'$ where $a \models \alpha$ and $X' \equiv \hat{X}'$. Therefore $\lambda_e \xrightarrow{!;\alpha} \lambda'_e$ and $(\hat{X}', \lambda'_e) \in \hat{\mathfrak{R}}$. By the definitions of $\hat{P}_e^{must}$ and $\hat{t}_e^{must}$, $l_e \xrightarrow{a} l'_e$, where $l'_e \models \lambda'_e$. Thus $(X', l'_e) \in \mathfrak{R}$.
   - Assume $X \xrightarrow{*a^1} X^1 \xrightarrow{*a^2} \cdots \xrightarrow{*a^k} X^k$, $X \xrightarrow{*b} X'$ and $l_e \xrightarrow{a^1} l_e^1 \xrightarrow{a^2} \cdots \xrightarrow{a^k} l_e^k \xrightarrow{b} l_e^{k+1}$. We have to show that $(X \cup X^k, l_e^k) \in \mathfrak{R}$. By the definitions of $\hat{P}^{may}$, $\hat{P}_e^{may}$, $\hat{t}^{may}$ and $\hat{t}_e^{may}$ we have that $\hat{X} \xrightarrow{?;*\alpha^1} \hat{X}^1 \xrightarrow{?;*\alpha^2} \cdots \xrightarrow{?;*\alpha^k} \hat{X}^k$, $\hat{X} \xrightarrow{?;*b} \hat{X}'$ and $\lambda_e \xrightarrow{?;\alpha^1} \lambda_e^1 \xrightarrow{?;\alpha^2} \cdots \xrightarrow{?;\alpha^k} \lambda_e^k \xrightarrow{?;b} \lambda_e^{k+1}$. Therefore $(\hat{X} \cup \hat{X}^k, \lambda_e^k) \in \hat{\mathfrak{R}}$. Since $l_e^k \models \lambda_e^k$ and $X \cup X^k \equiv \hat{X} \cup \hat{X}^k$ it follows that $(X \cup X^k, l_e^k) \in \mathfrak{R}$.

   Consequently $\mathfrak{R}$ is an agent-environment simulation between T and e.

2. Suppose that $(\hat{T} \leq \hat{e}) = \mathrm{ff}$. Then there is no relation $\hat{\mathfrak{R}} \subseteq \mathcal{P}(\hat{L}) \times \hat{L}_e$ such that $(\{\hat{\iota}\}, \hat{\iota}_e) \in \mathfrak{R}$ and whenever $\hat{X} \sim \lambda_e$ the following conditions hold:

   (a) If $\hat{X} \xrightarrow{!;*\alpha} \hat{X}'$, then $\lambda_e \xrightarrow{?;\alpha} \lambda'_e$ and $(\hat{X}', \lambda'_e) \in \hat{\mathfrak{R}}$.

   (b) If $\hat{X} \xrightarrow{!;*\alpha^1} \hat{X}^1 \xrightarrow{!;*\alpha^2} \cdots \xrightarrow{!;*\alpha^k} \hat{X}^k$, $\hat{X} \xrightarrow{!;*b} \hat{X}'$ and $\lambda_e \xrightarrow{!;\alpha^1} \lambda_e^1 \xrightarrow{!;\alpha^2} \cdots \xrightarrow{!;\alpha^k} \lambda_e^k \xrightarrow{!;b} \lambda_e^{k+1}$, then $(\hat{X} \cup \hat{X}^k, \lambda_e^k) \in \hat{\mathfrak{R}}$.

   So there must be a sequence

   $$\{\hat{\iota}\} \xrightarrow{!;\alpha_1} \hat{X}^1 \xrightarrow{!;\alpha_2} \cdots \xrightarrow{!;\alpha_r} \hat{X}^r$$

   for some $r \geq 1$ such that for all sequences

   $$\hat{\iota_e} \xrightarrow{?;\alpha_1} \lambda_e^1 \xrightarrow{?;\alpha_2} \cdots \xrightarrow{?;\alpha_r} \lambda_e^r,$$

   either one of the following holds:

   (i) $\hat{X}^r \xrightarrow{!;\alpha^{r+1}} \hat{X}^{r+1}$ and $\alpha \notin \hat{P}_e^{may}(\lambda_e^r)$.

   (ii) $\hat{X}^r \xrightarrow{!;*\alpha^{r+1}} \hat{X}^{r+1} \xrightarrow{!;*\alpha^{r+1}} \cdots \xrightarrow{!;*\alpha^{r+k}} \hat{X}^{r+k}$, $\hat{X}^r \xrightarrow{!;*b} \hat{X}'$ and $\lambda_e^r \xrightarrow{!;\alpha^{r+1}} \lambda_e^{r+1} \xrightarrow{!;\alpha^{r+2}} \cdots \xrightarrow{!;\alpha^{r+k}} \lambda_e^{r+k} \xrightarrow{!;b} \lambda_e^{r-}$
   then $(\hat{X} \cup \hat{X}^k, \lambda_e^k) \in$
   $\hat{X}_k \xrightarrow{!;\alpha} \hat{X}$, $\hat{X}_k \xrightarrow{!;\beta} \hat{X}'$, $\hat{l}_{ek} \xrightarrow{!;\alpha} \lambda_e \xrightarrow{!;\beta} \lambda'_e$ and $\alpha \notin \hat{P_e}^{may}(\lambda_e)$.

   (ii) $\hat{X}_k \xrightarrow{!;\alpha} \hat{X}$, $\hat{X}_k \xrightarrow{!;\beta} \hat{X}'$, $\hat{l}_{ek} \xrightarrow{!;\alpha} \lambda_e \xrightarrow{!;\beta} \lambda'_e$ and $\alpha \notin \hat{P_e}^{may}(\lambda_e)$.

   We have to show that $T \not\leq e$. For contradition suppose that $T \leq e$. Assume (i) above; the case of (ii) is similarly shown. By the definitions of $\hat{P}^{must}$ and $\hat{t}^{must}$ there is a sequence

   $$\iota \xrightarrow{a_1} X_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{k-1}} X_k \xrightarrow{a} X.$$

   As $T \leq e$ there must be a sequence

   $$\iota_e \xrightarrow{a_1} l_{e1} \xrightarrow{a_2} \cdots \xrightarrow{a_{k-1}} l_{ek} \xrightarrow{a} l_e.$$

   As $a \models \alpha$ and $\alpha \notin \hat{P}_e^{may}(\lambda_{ek})$ we have that $a \notin P_e(l_{ek})$. Thus $l_{ek} \xrightarrow{a} l_e$ does not hold. The latter is a contradition.

□

Theorem 5 (PMC theorem)**.** *Let $\mathcal{S} = \langle T, e, \mathcal{V} \rangle$ be a PIIS^{UV} with predicate abstraction $\hat{\mathcal{S}} = \langle \hat{T}, \hat{e}, \hat{\mathcal{V}} \rangle$ such that $(\hat{T} \leq \hat{e}) = \mathrm{tt}$. Let ϕ be an m-indexed formula. The following hold:*

1. $(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{tt} \Longrightarrow \mathcal{S} \models^2 \phi$.

2. $(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{ff} \Longrightarrow \mathcal{S} \not\models^2 \phi$.

Proof. From Theorem 2

$$(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{tt} \Longrightarrow \mathcal{S}(m) \models^2 \phi$$

and

$$(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{ff} \Longrightarrow \mathcal{S}(m) \not\models^2 \phi.$$

By assumption and Theorem 4 we have that $T \leq e$. Thus Theorem 3 gives

$$\mathcal{S}(m) \models^2 \phi \Longrightarrow \mathcal{S} \models^2 \phi$$

and

$$\mathcal{S}(m) \not\models^2 \phi \Longrightarrow \mathcal{S} \not\models^2 \phi.$$

Consequently

1. $(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{tt} \Longrightarrow \mathcal{S} \models^2 \phi$.
2. $(\hat{\mathcal{S}}(m) \models^3 \phi) = \mathrm{ff} \Longrightarrow \mathcal{S} \not\models^2 \phi$.

□