Parameterised Verification of Data-aware Multi-agent Systems

Francesco Belardinelli
Laboratoire IBISC, UEVE
IRIT Toulouse, France
belardinelli@ibisc.fr

Panagiotis Kouvaros
Department of Computing
Imperial College London, UK
Univ. of Naples “Federico II”, Italy
p.kouvaros@imperial.ac.uk

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

Abstract

We introduce parameterised data-aware multi-agent systems, a formalism to reason about the temporal properties of arbitrarily large collections of homogeneous agents, each operating on an infinite data domain. We show that their parameterised verification problem is semi-decidable for classes of interest. This is demonstrated by separately addressing the unboundedness of the number of agents and the data domain. In doing so we reduce the parameterised model checking problem for these systems to that of parameterised verification for interleaved interpreted systems. We illustrate the expressivity of the formal model by modelling English auctions with an unbounded number of bidders on unbounded data.

1 Introduction

There has been recent interest in the study of data-aware multi-agent systems (DAMAS) [Montali *et al.*, 2014; Belardinelli *et al.*, 2014; Calvanese *et al.*, 2016]. While standard multi-agent systems (MAS) are modelled by studying the properties of the underlying processes or *agents*, in DAMAS the emphasis is given equally to the agents and the *data* driving the executions. This paradigm shift answers a growing demand from applications to support fully the ever increasing amount of data generated by and available to current networked applications [Singh and Huhns, 2005]. A successful paradigm in data-aware systems is that of artifact-centric systems, which have been used to model and execute, among others, data-aware services [De Masellis *et al.*, 2015], hierarchical systems [Deutsch *et al.*, 2016], and case-centric applications [Montali and Calvanese, 2016].

Verifying DAMAS is challenging because of the infinite state models generated by their infinite-domain variables. Approaches based on abstraction have been put forward to solve this problem [Lomuscio and Michaliszyn, 2014; Belardinelli *et al.*, 2014; Montali and Calvanese, 2016], and related techniques have been suggested for similar systems [Bagheri *et al.*, 2013; Gonzalez *et al.*, 2012]. While these investigations have resulted in sound methodologies and open-source toolkits [Gonzalez *et al.*, 2015], a key limitation of DAMAS is that the number of agents in the system is fixed and given at design-time. This is in marked contrast with the range of applications DAMAS are meant to be employed for (services, case-management, auction-based mechanisms, etc.), which precisely rely on the fact that data-centric structures interact with an unbounded number of actors.

In this contribution we address this issue by introducing parameterised data-aware MAS (or P-DAMAS) as systems with an unbounded number of homogenous agents, each assumed to be *data-aware*, i.e., endowed with possibly infinite domains and interacting with an environment composed of partially shared data. Specifically, we here tackle the question of verifying P-DAMAS against MAS-oriented specifications. Since the latter involve quantification over data, we combine temporal logic together with first-order features. To deal with the infinity arising from infinite-state variables, we use abstraction techniques based on simulations. To overcome the problems arising from an unbounded number of agents, we develop a parameterised verification technique. The key contribution shows that the verification of particular classes of P-DAMAS that we introduce is partially decidable.

The rest of the paper is organised as follows. In Section 2 we introduce the syntax and semantics of P-DAMAS. In Section 3 we identify a class of P-DAMAS for which we give the semi-decidability result. We illustrate the method in Section 4, where we discuss the verification of auction-based mechanisms. All proofs are omitted for reasons of space.

Related Work. As mentioned above, several proposals have been put forward to verify DAMAS and artifact-centric systems, including [Belardinelli *et al.*, 2012; Lomuscio and Michaliszyn, 2014; Belardinelli *et al.*, 2014; Montali and Calvanese, 2016; Bagheri *et al.*, 2013; Belardinelli and Lomuscio, 2016]. None of these approaches deals with an unbounded number of agents as we do here. Methods for the verification of unbounded MAS have also been developed [Kouvaros and Lomuscio, 2013a; 2016; 2015a; 2015b]; however, these technique do not deal with infinite-state agents. More recently a method for the verification of parameterised MAS, each encoded via infinite-state models, was suggested [Kouvaros and Lomuscio, 2017]. However, the approach targets a non-quantified specification language and does not deal with (semi-)structured data as we do here. As a result, their method differs from the one we present and it is applicable to an uncomparable class of systems.

2 Parametric Data-aware MAS

We introduce parametric data-aware multi-agent systems (P-DAMAS) an extension of infinite-state reactive modules [Belardinelli and Lomuscio, 2016], where the number of agents is unbounded. P-DAMAS consist of an *agent template*, from which an unbounded number of homogeneous (concrete) agents may be constructed, as well as an *environment* in which the agents operate. The agent template and the environment admit variables with an infinite domain of interpretation, possibly totally ordered (e.g., natural numbers). The specifications for these systems will be given in a parametric first-order extension of the branching-time temporal logic CTL [Clarke *et al.*, 1999].

Agent templates. In the following we assume an ordered *interpretation domain* D, a set $\mathit{var} = \{v_0, v_1, \ldots\}$ of *variables* and a set $\mathit{par} = \{x_0, x_1, \ldots\}$ of *parameters* interpreted on D. Variables are used to describe the data model, while parameters appear in formulas. Further, we introduce an *agent template* t (or simply template) as well as the environment e. In line with reactive systems [Alur and Henzinger, 1999], we assume that each $i \in \{t, e\}$ controls a *finite* set $cnt_i \subseteq \mathit{var}$ of variables. Specifically, $\{cnt_e, cnt_t\}$ forms a partition of *var*. Hence, the set *var* can be assumed to be finite. The set obs_t of variables that are observable by agent template t includes all of her controlled variables as well as the variables controlled by the environment: $obs_t = cnt_t \cup cnt_e$.

In line with the formal account of agents in the literature on interpreted systems [Fagin *et al.*, 1995], we suppose each $i \in \{t, e\}$ has a set L of *local states*, a set Act of *actions*, and a *protocol function* P. In particular, to introduce a formal account of local state, we consider *local interpretations* as functions $\theta_i : cnt_i \to D$, i.e., (finite) assignments from the variables in cnt_i to values in D. For simplicity, we often identify an interpretation θ_i with its range $\theta_i(cnt_i) \subseteq D$, whenever domain cnt_i is clear by the context. Then, a local state $l \in L$ of agent template t includes all values of its observed variables in obs_t, i.e., $l = \theta_t \cup \theta_e$. Since the domain D is infinite in general, the set L of local states is also infinite.

To define the individual actions in Act and the protocol P, we introduce a first-order language built on variables, parameters and relation symbols $=$ and $\leqslant$.

Definition 1 (FO-formulas)**.** *First-order formulas are defined according to the following BNF, where* $z, z' \in \mathit{var} \cup \mathit{par}$ *and* $x \in \mathit{par}$*:* $\phi ::= z = z' \mid z \leqslant z' \mid \neg\phi \mid \phi \to \phi \mid \forall x \phi$

The symbols $\neq$, $<$, $\geqslant$, $\top$, $\bot$, connectives $\wedge$, $\vee$, quantifier $\exists$, and free and bound variables and parameters are defined as standard [Hamilton, 1978]. Notice that quantification applies to parameters only, this is in accordance with the intuition above on the use of variables and parameters.

Definition 2 (Guarded Command)**.** *A* guarded command γ *over var and par is an expression*

$$id \;\; \equiv \;\; g(x_1, \ldots, x_k) \rightsquigarrow v_1 := x_1; \ldots; v_k := x_k$$

where (i) *id is the command's identity;* (ii) guard *g is an FO-formula with free parameters among* $x_1, \ldots, x_k$.

The intuitive meaning of a guarded command is that if guard g is true for some interpretation $\sigma : \mathit{par} \to D$ of parameters, then the command γ is enabled for execution. By executing γ we set each variable v_i to value $\sigma(x_i) \in D$. In particular, the *skip* command can be represented as $\top \rightsquigarrow \epsilon$, where ϵ is the empty sequence. We say that $v_1, \ldots, v_k$ are the variables *controlled* by γ, and denote this set by $ctr(\gamma)$, while the variables in g are the *observable* variables $obs(\gamma)$ [Hoek *et al.*, 2006].

Following the typical setting in parameterised formalisms for MAS [Kouvaros and Lomuscio, 2013b; 2016], we assume that each command can either be an *asynchronous command*, an *agent-environment command*, or a *global-synchronous command*. Each type of command enables a different communication pattern between the concrete agents instantiated from the templates. Specifically, asynchronous commands enable the asynchronous evolution of an agent; agent-environment commands enable pairwise synchronisation between one agent and the environment; global-synchronous commands enable full synchronisation among all the agents and the environment.

To introduce the semantics of guarded commands formally, we define the satisfaction $\models$ of FO-formulas. An FO-formula ϕ is given meaning by a *finite interpretation* $\sigma : fr(\phi) \to D$ that assigns values in D to the free parameters in ϕ. A *reinterpretation* σ^x_u coincides with σ, but assigns value $u \in D$ to parameter $x \in fr(\phi)$. Given $z \in \mathit{var} \cup \mathit{par}$, $(\theta, \sigma)(z) = \theta(z)$ for $z \in \mathit{var}$, and $(\theta, \sigma)(z) = \sigma(z)$ for $z \in \mathit{par}$.

Definition 3 (Satisfaction)**.** *The satisfaction of an FO-formula* ϕ *for a finite interpretation* σ *and local interpretation* θ*, denoted* $(\theta, \sigma) \models \phi$*, is defined as follows (clauses for propositional connectives are immediate and thus omitted):*

$$\begin{array}{lll}
(\theta, \sigma) \models z = z' & \textit{iff} & (\theta, \sigma)(z) = (\theta, \sigma)(z') \\
(\theta, \sigma) \models z \leqslant z' & \textit{iff} & (\theta, \sigma)(z) \leqslant (\theta, \sigma)(z') \\
(\theta, \sigma) \models \forall x \phi & \textit{iff} & \textit{for all } u \in \theta(\mathit{var}),\ \sigma^x_u \models \phi
\end{array}$$

The interpretation of FO-formulas is completely standard, but for quantification that takes values from the finite image $\theta(\mathit{var}) = \{u \in D \mid u = \theta(v) \text{ for some } v \in \mathit{var}\}$ of *var*. This is consistent with the interpretation of quantification on *active domains* in database theory [Abiteboul *et al.*, 1995]. Indeed, at this stage quantification can be considered syntactic sugar, as $\theta(\mathit{var})$ is finite.

Definition 4 (Agent template)**.** *The* agent template *is a tuple* $t = \langle L, init, Act, P, \tau \rangle$ *where*

- $L = \{\theta_t \cup \theta_e \mid \theta_i : ctr_i \to D \textit{ for } i \in \{t, e\}\}$*, where* ctr_t *and* ctr_e *are the (finite) set of variables owned by t, e;*
- $init = \iota_t \cup \iota_e$*, where* $\iota_t : ctr_t \to D$ *and* $\iota_e : ctr_e \to D$ *provide the initial interpretations of* ctr_t *and* ctr_e*;*
- *Act is a (infinite) set of pairs* $\alpha = (\gamma, \sigma)$ *of guarded commands* γ*, together with finite interpretations* σ*, s.t. for every* γ*,* $ctr(\gamma) \subseteq ctr_t$ *and* $obs(\gamma) \subseteq obs_t$*;*
- $P : L \to \wp(Act) \backslash \{\varnothing\}$ *is such that, for every* $l \in L$*,* $P(l) = \{\alpha \in Act \mid (l, \sigma_\alpha) \models \gamma_\alpha\}$*;*
- $\tau : L \times Act \times Act_e \to L$ *is such that* (i) $\tau(l, \alpha, \alpha_e)$ *is defined only if* $\alpha \in P(l)$*; and* (ii) $\tau(l, \alpha, \alpha_e) = l'$ *iff for every variable* $v_i \in cnt(\gamma_\alpha)$ *and* $w_i \in cnt(\gamma_{\alpha_e})$*,* $\theta'(v_i) = \sigma_\alpha(x_i)$ *and* $\theta'_e(w_i) = \sigma_{\alpha_e}(y_i)$*; while all other variables do not change value.*

The environment is similarly defined. Observe, however, that its set of local states is defined only on θ_e and its transition function is defined only on its current state l_e and action α_e. The actions of the agent template are partitioned as $Act = A \cup AE \cup GS$, where A is a set of asynchronous actions, AE is a set of agent-environment actions, and GS is a set of global-synchronous actions. Concretely, the agents synchronise on actions with the same identity. Given a set X of actions, let $id(X) = \{id_\gamma \mid (\gamma, \sigma) \in X\}$ be the set of the commands' identities in X. Following the agent-environment and global-synchronous synchronisation patterns we assume that $id(AE_e) = id(AE_t)$ and $id(GS_t) = id(GS_e)$.

Finally, a parametric data-aware multi-agent system is a pair of an agent template and an environment.

Definition 5 (P-DAMAS)**.** *A* parametric data-aware multi-agent system *(P-DAMAS) is a pair $\mathfrak{M} = \langle t, e\rangle$, where t is the agent template and e is the environment.*

P-DAMAS provide a description of an unbounded collection of (concrete) *data-aware multi-agent systems* (DAMAS).

Concrete Agents. Concrete DAMAS are obtained by setting the parameters to the actual number of agents in the system. That is, given a P-DAMAS $\mathfrak{M}$ and $n \in \mathbb{N}$, the DAMAS $M(n)$ of n agents per template t is the composition of n copies of t with the environment. We write $Ag(n) = \{t_j \mid 1 \leqslant j \leqslant n\}$ for the set of all concrete agents $t_j = \langle L_j, Act_j, init_j, P_j, \tau_j\rangle$. The concrete agent inherits from the template her actions, her protocol, and her transition function. However, these are defined on variables that are indexed by the agent's identity. Specifically, we consider the set $var(n) = \{v \times \{1, \ldots, n\} \mid v \in ctr_t\} \cup ctr_e$ of variables, where agent t_j controls the variables in $ctr_j = \{v_j \in var(n) \mid v \in ctr_t\}$ and observes the variables in $obs_j = ctr_j \cup ctr_e$. This is consistent with the requirement that $\{ctr_1, \ldots, ctr_n, ctr_e\}$ form a partition of $var(n)$.

Definition 6 (Concrete agent)**.** *Given the agent template $t = \langle L, init, Act, P, \tau\rangle$, the j-th concrete agent instantiated from t is a tuple $t_j = \langle L_j, init_j, Act_j, P_j, \tau_j\rangle$, where*

- *$L_j = \{\theta_j \cup \theta_e \mid \theta_i : ctr_i \to D$ for $i \in \{j, e\}\}$;*
- *$init_j = \iota_j \cup \iota_e$, where $\iota_j : ctr_j \to D$ is such that $\iota_j(v_j) = x$ iff $\iota_t(v) = x$;*
- *$Act_j = \{(\gamma', \sigma) \mid (\gamma, \sigma) \in Act_t\}$, where γ' is obtained from γ by replacing every variable $v \in ctr(\gamma)$ by v_j;*
- *The protocol P_j and the transition function τ_j are defined as in Def. 4.*

Def. 6 above provides the concrete counterpart to the notion of agent template introduced in Def. 4. Further, a *global state* in DAMAS $M(n)$ is a tuple $s = \langle\theta_1, \ldots, \theta_n, \theta_e\rangle$, where each $\theta_j : cnt_j \to D$ is an interpretation for the j-th instantiation of template t. Equivalently, global states can be represented as functions $s : var(n) \to D$, i.e., finite interpretations of the variables in $var(n)$ with values in D such that for every $v_j \in var(n)$, $s(v) = \theta_j(v)$. As anticipated above, any state s is well-defined as $var(n)$ is partitioned among the agents in $Ag(n)$. Further, given a global state s, we denote as $l_1, \ldots, l_n, l_e$ the corresponding local states for all agents in $Ag(n) \cup \{e\}$. Observe that $\langle\theta_1, \ldots, \theta_n, \theta_e\rangle$ and $\langle l_1, \ldots, l_n, l_e\rangle$ are equivalent representations of global state s, in terms of controlled, respectively observable, variables. So, we will use the two notations interchangeably. We stress that concrete agents have only partial observability of the global state of the system.

Let $ACT = \prod_{ag \in Ag(n) \cup \{e\}} Act_{ag}$ be the set of joint actions. For $\overline{a} \in ACT$, consider $\overline{\alpha}.ag$ to represent the action of agent ag. The concrete system evolves over time in compliance with the agents' protocols and evolution functions. This is described by the global transition function.

Definition 7 (Global transition function)**.** *The global transition function $\tau : \mathcal{G} \times ACT \to \mathcal{G}$ is defined as follows: $\tau(s, \overline{\alpha}) = s'$ iff for every $ag \in Ag(n)$, $l'_{ag} = \tau_{ag}(l_{ag}, \overline{\alpha}.ag, \overline{\alpha}.e)$, $l'_e = \tau_e(l_e, \overline{\alpha}.e)$, and one of the following holds:*

- *(Asynchronous): for some $ag \in Ag(n)$, (i) $\overline{\alpha}.ag$ is asynchronous; and (ii) for every $ag' \neq ag$, $\overline{\alpha}.ag' = skip$.*
- *(Agent-environment): for some $ag \in Ag(n)$, (i) $\overline{\alpha}.ag$ is an agent-environment action; (ii) $id_{\overline{\alpha}.e} = id_{\overline{\alpha}.ag}$; and (iii) for every $ag' \neq ag$, $ag' \neq e$, $\overline{\alpha}.ag' = skip$.*
- *(Global-synchronous): for every $ag, ag' \in Ag(n) \cup \{e\}$, (i) $\overline{\alpha}.ag$ is a global-synchronous action; and (ii) $id_{\overline{\alpha}.ag} = id_{\overline{\alpha}.ag'}$.*

Above τ defines only one action to be performed at each time step. If this is an asynchronous action, then exactly one concrete agent participates in the global transition; if it is an agent-environment action, then exactly one concrete agent and the environment participate in the transition; if it is a global-synchronous action, then all concrete agents and the environment participate in the transition. The agents not participating in the transition are assumed to perform the *skip* action. Moreover, by the definition of each τ_{ag}, we have that for every $ag \in Ag(n) \cup \{e\}$, $\overline{a}.ag \in P_{ag}(s_{ag})$.

We can now define the concrete systems generated from a P-DAMAS $\mathfrak{M}$.

Definition 8 (DAMAS)**.** *The* data-aware MAS *(DAMAS) $M(n)$ of n agents is a tuple $M(n) = \langle S, init, \tau\rangle$, where: $init = \prod_{ag \in Ag(n) \cup \{e\}} init_{ag}$; τ is the global transition function (Definition 7); S is the closure of $init$ according to τ.*

Clearly, a P-DAMAS generates different DAMAS depending on the number n of agents in the system. Overall, a concrete DAMAS $M(n)$ describes the evolution of a multi-agent system from the initial state $init$, according to the transition function τ. Again, since the domain D is infinite in general, every generated DAMAS is an infinite-state system.

The Specification Language. To reason about an unbounded number of agents, we here define an indexed, first-order extension of the temporal logic ECTL$\backslash X$ (the existential fragment of CTL without next X), where the atomic propositions are indexed by *agent parameters*. These are agent-specific parameters whose domain depends on the concrete system on which the specification is evaluated: if it is evaluated on $M(n)$, then the potential set of values is $\{1, \ldots, n\}$. For agent template t consider a set *apar* of *agent parameters*. Intuitively, indexed formulas quantify universally over the concrete agents.

Definition 9 (Indexed FO-formulas and FO-ECTL$\backslash X$)**.** *Indexed first-order formulas over agent parameters apar are defined according to the following BNF, where $z = (v, a)$*

(resp. $z' = (v', a')$), for $v, v' \in$ ctr, $a, a' \in$ apar, and $x \in$ par: $\phi ::= z = z' \mid z \leqslant z' \mid \neg\phi \mid \phi \rightarrow \phi \mid \forall x \phi$

Formulas in first-order ECTL$\backslash X$ are defined as follows, where ϕ is an indexed FO-formula:

$$\psi \;::=\; \phi \mid \neg\phi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \forall x \psi \mid E(\psi U \psi) \mid E(\psi R \psi) \mid \forall^{ag} a \psi$$

Note that we use $\forall^{ag}$ to indicate that the operator quantifies over agent parameters. The temporal modality $E(\phi U \psi)$ stands for "for some path, ϕ holds until ψ holds"; and $E(\phi R \psi)$ denotes "for some path, ϕ releases ψ". We say that an FO-ECTL$\backslash X$ sentence is *m-indexed*, for $m \in \mathbb{N}$, if there are precisely m agent parameters from *apar* appearing in the formula. Notice that in FO-ECTL$\backslash X$ we can have arbitrary alternations of quantifiers and ECTL$\backslash X$ operators. A consequence of this is that quantification in FO-ECTL$\backslash X$ is not syntactic sugar.

We now define the satisfaction relation. In the definition we assume that the sets of parameters appearing in the commands and the formula are disjoint. This can be done without loss of generality, as both sets are finite and defined at design-time. Hereafter a *path* is an infinite sequence $\pi = s^1 \overline{\alpha}^1 s^2 \overline{\alpha}^2 s^3 \ldots$ with $\tau(s^i, \overline{\alpha}^i) = s^{i+1}$, for every $i \geqslant 1$. Given a path π, we write $\pi(i)$ for the i-th state in π. The set of all paths originating from a state s is denoted by *Path*(s).

Definition 10 (Satisfaction). *The satisfaction relation $\models$ for a DAMAS $M(n)$, a global state s, an FO-ECTL$\backslash X$ formula ψ, and an interpretation σ is defined as follows (clauses for propositional connectives are immediate and thus omitted).*

$$\begin{array}{lll}
(M(n), s, \sigma) \models \phi & \textit{iff} & (s, \sigma) \models \phi, \textit{ where } \phi \textit{ is an FO-formula} \\
(M(n), s, \sigma) \models \forall x \psi & \textit{iff} & \textit{for all } u \in s(\textit{var}(n)),\ (M, s, \sigma^x_u) \models \psi \\
(M(n), s, \sigma) \models E(\psi U \psi') & \textit{iff} & \textit{for some } \pi \in \textit{Path}(s), \textit{ for some } i \geqslant 0, \\
 & & (M(n), \pi(i), \sigma) \models \psi' \textit{ and for all } j < i, \\
 & & (M(n), \pi(j), \sigma) \models \psi \\
(M(n), s, \sigma) \models E(\psi R \psi') & \textit{iff} & \textit{for some } \pi \in \textit{Path}(s), \textit{ either for some } i \geqslant \\
 & & 0,\ (M(n), \pi(i), \sigma) \models \psi \textit{ and for all } j \leqslant i \\
 & & (M(n), \pi(j), \sigma) \models \psi', \textit{ or for all } i \geqslant 0 \\
 & & (M(n), \pi(i), \sigma) \models \psi' \\
(M(n), s, \sigma) \models \forall^{ag} y \psi & \textit{iff} & ag \in \{1, \ldots, n\} \textit{ implies} \\
 & & (M(n), s', \sigma) \models \psi[y \mapsto ag], \textit{ for} \\
 & & ag \in apar
\end{array}$$

We remark that the semantics of ECTL$\backslash X$ operators in Def. 10 is standard, while quantification over regular parameters ranges on the active domain $s(\textit{var}(n))$. However, differently from Def. 3, quantification is not syntactic sugar: transitions might take us to a successor state s', in which an individual $u \in s(\textit{var}(n))$ is no longer active, i.e., $u \notin s'(\textit{var}(n))$. As a result, quantification in FO-ECTL$\backslash X$ gives us a language that is strictly more expressive than propositional ECTL$\backslash X$.

An FO-ECTL$\backslash X$ formula ψ is *true* in state s, or $(M(n), s) \models \psi$, iff for all interpretations σ, $(M(n), s, \sigma) \models \psi$; ψ is *true* in $M(n)$, or $M(n) \models \psi$, iff $(M(n), \textit{init}) \models \psi$. In light of decidability limitations (see [Bloem *et al.*, 2015] for a detailed discussion), hereafter we consider *prenex m*-indexed FO-ECTL$\backslash X$ formulas in which the universal quantifiers on *apar* appear only at the front of the formula.

We can now state the parameterised model checking problem for the present setting.

Definition 11 (PMCP for P-DAMAS). *Given a P-DAMAS $\mathfrak{M}$ and an m-indexed FO-ECTL$\backslash X$ formula ψ, the* parameterised model checking problem *consists in determining whether for all $n \geqslant m$, $M(n) \models \psi$.*

Parameterised model checking involves checking an unbounded number of systems. Since P-DAMAS extend broadcast protocols whose PMCP is undecidable [Esparza *et al.*, 1999], the PMCP the P-DAMAS is also undecidable. general [Apt and Kozen, 1986]. Moreover, notice that each concrete system is an infinite-state system, and again the model checking problem for infinite-state systems is normally undecidable [Deutsch *et al.*, 2009]. However, in what follows we define a cutoff technique to bound the number of agents to check, thereby obtaining partial decidability.

3 Partial Decidability via Abstractions

In this section we develop a partial model checking procedure for FO-ECTL$\backslash X$. Specifically, the partial decidability of the parameterised verification problem is given in two steps. In the first step, the domain D of the P-DAMAS to be verified is abstracted into a finite domain D^A. It is shown that every concrete system generated from the abstract P-DAMAS defined on D^A is simulated by the equally populated concrete system obtained from the original P-DAMAS built on D. As a result, the PMCP is reduced to checking an unbounded number of finite-state systems. In the second step, a mapping is defined from (abstract) finite state P-DAMAS to parameterised interleaved interpreted systems (PIIS) [Kouvaros and Lomuscio, 2016]. Consequently, we can apply the results in [Kouvaros and Lomuscio, 2016] to solve the PMCP.

Finite Simulations. First of all, notice that Def. 4 of agent template depends on the interpretation domain D as well. That is, by varying D we can obtain P-DAMAS defined on the same partition of variables, but with different interpretations. In particular, if $D^A \subseteq D$ is finite, then the corresponding P-DAMAS is finite as well, and while we can still have an unbounded number of agents in the concrete DAMAS, each DAMAS itself is a finite-state system. Hereafter we prove that, whenever $D^A \subseteq D$, for every $n \in \mathbb{N}$, the concrete, possibly finite DAMAS $M^A(n)$ built on D^A is a submodel of the concrete, infinite-state DAMAS $M(n)$ defined on D. In particular, the former is simulated by the latter. As a consequence, existential formulas in FO-ECTL$\backslash X$ are preserved from $M^A(n)$ to $M(n)$.

Definition 12 (Abstract Template and Abstract P-DAMAS). *Let $i \in \{t, e\}$ be an agent template (resp. the environment) whose controlled variables in cnt_i take values in domain D, and let $D^A \subseteq D$. Then the* abstraction *i^A is obtained by restricting the range of variables in cnt_i to D^A.*

Further, given P-DAMAS $\mathfrak{M}^A = \langle t, e \rangle$, the abstract *P-DAMAS $\mathfrak{M} = \langle t^A, e^A \rangle$ is the collection of abstractions t^A and e^A built on D^A.*

Given $n \in \mathbb{N}$, the DAMAS $M^A(n)$ for n agents per abstract template t^A is defined as the composition of n copies of t^A with the abstract environment e^A, in analogy with Def. 6 and 7. In particular, observe that if s is a state in DAMAS $M^A(n)$, then s also belongs to the concrete $M(n)$. Hence, $M^A(n)$ is a submodel of $M(n)$. In particular, $M(n)$ simulates $M^A(n)$. To prove this fact we state some partial results.

Lemma 1. *For every states s, s' in $M^A(n)$ and joint action $\alpha \in ACT$, if $s \xrightarrow{\alpha} s'$ in $M^A(n)$, then $s \xrightarrow{\alpha} s'$ in $M(n)$.*

By Lemma 1 all transitions in $M^A(n)$ are simulated in $M(n)$. This result can be extended to whole paths.

Lemma 2. *Every path π from s in $M^A(n)$ is also a path (from s) in $M(n)$.*

By Lemma 2 we can prove the main preservation result of this section.

Theorem 3. *Let $M(n)$ be a DAMAS with abstraction $M^A(n)$ defined on $D^A \subseteq D$. For every states s in $M^A(n)$ and formula ϕ in FO-ECTL$\backslash X$, if $M^A(n) \models \phi$, then $M(n) \models \phi$.*

In particular, by Theorem 3 existential formulas are preserved by taking DAMAS defined on a finite domain $D^A \subseteq D$. However, in principle we have an infinite number of such finite DAMAS $M(n)$, one for every choice of agent parameter n. We tackle this issue in the following section.

PIIS simulations. We reduce the PMCP for finite-state P-DAMAS to the PMCP for PIIS. That is, we show that for every abstract P-DAMAS $\mathfrak{M}^A$ we can associate a PIIS $\mathfrak{M}^{PA}$ whose concrete systems satisfy the same FO-ECTL$\backslash X$ formulas as the equally populated concrete systems from $\mathfrak{M}^A$. Recall that PIIS are defined as finite-state P-DAMAS, but with the following differences: (i) the variables controlled by the environment are private to the environment, i.e., $obs_t = cnt_t$; (ii) the agent template's transition function does not depend on the action of the environment, i.e., $\tau : L \times Act \rightarrow L$. Accounting for these differences we now define $\mathfrak{M}^{PA}$. We begin with the definition of the notions of *guarded command products* and *AE-synchronisation commands*. Intuitively, the commands enable the PIIS agents to simulate the updates of the observable components of the DAMAS agents' states.

Definition 13 (Guarded command products)**.** *The* product *of two guarded commands $id \equiv g(x_1, \ldots, x_k) \rightsquigarrow v_1 := x_1; \ldots; v_k := x_k$ and $id \equiv g'(x'_1, \ldots, x'_{k'}) \rightsquigarrow v'_1 := x'_1; \ldots; v'_{k'} := x'_{k'}$ is defined as the guarded command $id \equiv g(x_1, \ldots, x_k) \wedge g'(x'_1, \ldots, x'_{k'}) \rightsquigarrow v_1 := x_1; \ldots; v_k := x_k; v'_1 := x'_1; \ldots; v'_{k'} := x'_{k'}$.*

The product of an agent's command and the environment's command enables a PIIS agent to explicitly update the environment's variables encoded in the agent's state. Given actions $a = (\gamma, \sigma)$, $a' = (\gamma', \sigma')$, we write $a \times a' = (\gamma \times \gamma', \sigma \cup \sigma')$ for their product.

Definition 14 (AE synchronisation commands)**.** *Let γ be an agent-environment command $id \equiv g(x_1, \ldots, x_k) \rightsquigarrow v_1 := x_1; \ldots; v_k := x_k$. The AE* initiator *command of γ, $\gamma[?]$, is the agent-environment command $id[?] \equiv g(x_1, \ldots, x_k) \wedge ae\_sync = \bot \rightsquigarrow ae\_sync = \top$. The AE broadcast command $\gamma[!]$ of γ is the global-synchronous command $id[!] \equiv g(x_1, \ldots, x_k) \wedge ae\_sync = \top \rightsquigarrow v_1 := x_1; \ldots; v_k := x_k; ae\_sync = \bot$.*

AE-synchronisation commands enable the PIIS agents to simulate the agent-environment transitions of the DAMAS agents. In particular the AE initiator command $\gamma[?]$ is performed by the agent participating in the agent-environment transition. The command "marks" said agent and signals the execution of the global-synchronous command $\gamma[!]$ by setting the (fresh) boolean variable ae_sync to $\top$. With the global synchronisation the agent updates both controlled and observable variables, whereas all other agents update only the observable variables (see item (ii) of Lemma 4).

We now define the PIIS $\mathfrak{M}^{PA}$ associated with $\mathfrak{M}^A$.

Definition 15 (Associated PIIS)**.** *The PIIS $\mathfrak{M}^{PA} = \langle t^{PA}, e^{PA} \rangle$ associated with P-DAMAS $\mathfrak{M}^A = \langle t^A, e^A \rangle$ over domain $D^A \cup \{ae\_sync\}$ is obtained from t^A, e^A by defining the following sets of actions for t^{PA} and e^{PA}:*

$Act_{t^{PA}}$: A is the set of asynchronous actions; $\{a[?] \mid a \in AE_t\}$ is the set of agent-environment actions; and $AE_e \cup \{a[!] \times a_e[!] \mid a \in AE_t, a_e \in AE_e, id_a = id_{a_e}\} \cup \{a \times a_e \mid a \in GS_t, a_e \in GS_e, id_a = id_{a_e}\}$ is the set of global-synchronous actions.

$Act_{e^{PA}}$: $\{a[?] \mid a \in AE_e\}$ is the set of agent-environment actions and $GS_e \cup \{a[!] \mid a \in GS_e\}$ is the set of global-synchronous actions.

Above we assume that ae_sync is initially set to $\bot$. Also, every action of t^{PA} that is not a broadcast action is guarded by the additional requirement that ae_sync is set to $\bot$. The following definition relates the states of each concrete system $M^A(n)$ to the states of the concrete system $M^{PA}(n)$.

Definition 16 (Related states)**.** *A global state s of $M^A(n)$ and a global state q of $M^{PA}(n)$ are* related*, or $s \approx q$, iff* (i) *for all $v \in avar(n)$, $s(v) = q(v)$; and* (ii) *for all $ag \in \{1, \ldots, n, e\}$, $s((ae\_sync), ag) = q((ae\_sync), ag) = \bot$.*

Following the above definition we show that related states satisfy the same FO-ECTL$\backslash X$ formulas. Since the initial states of corresponding concrete systems are related, the systems satisfy the same FO-ECTL$\backslash X$ formulas. To show this we first state some intermediate results.

Lemma 4. *Let s be a state of $M^A(n)$ and q a state of $M^{PA}(n)$. If $s \approx q$, then the following hold:*

(i) If $s \xrightarrow{\alpha} s'$, then $q \xrightarrow{\alpha} q'$ and $s' \approx q'$.

(ii) If $s \xrightarrow{\alpha} s'$ is an agent-environment transition fired by agent i, then $q \xrightarrow{\alpha'} q' \xrightarrow{\alpha''} q''$ and $s' \approx q''$, where α' is defined by $\alpha'.j = skip$ for $j \neq i \neq e$, $\alpha'.i = \alpha.i[?]$, and $\alpha'.e = \alpha.e[?]$; α'' is defined by $\alpha''.j = \alpha.e$ for all $j \neq i \neq e$, $\alpha''.i = \alpha.i[!] \times \alpha.e[!]$, $\alpha''.e = \alpha.e[!]$.

(iii) If $s \xrightarrow{\alpha} s'$ is a global synchronous transition, then $q \xrightarrow{\alpha'} q'$ and $s' \approx q'$, where α' is defined by $\alpha'.i = \alpha.i \times \alpha.e$ for $i \neq e$, and $\alpha'.e = \alpha.e$.

By Lemma 4 the transitions in $M^A(n)$ are simulated in $M^{PA}(n)$ [1]. Additionally, it is easy to see that transitions in $M^{PA}(n)$ are simulated in $M^A(n)$. We thus obtain the following preservation result.

Theorem 5. *Let $\mathfrak{M}$ be a P-DAMAS with abstraction $\mathfrak{M}^A$. Let $\mathfrak{M}^{PA}(n)$ be the PIIS associated with $\mathfrak{M}^A$. Then, for every formula ϕ in FO-ECTL$\backslash X$, $M^A(n) \models \phi$ iff $M^{PA}(n) \models \phi$.*

As a consequence, the PMCP for P-DAMAS can be solved by solving the PMCP for PIIS. Given an m-indexed formula, the latter problem can be solved by checking the concrete system with $\max(2, m)$ agents [Kouvaros and Lomuscio, 2013b]. The result is derived under the assumption that

[1] Note that since our specification logic does not include the next-time operator, a transition in $M^A(n)$ can be simulated by more than one transition in $M^{PA}(n)$ [Kouvaros and Lomuscio, 2016].

the environment is *non-blocking*. That is, whenever an agent-environment action, or a global synchronous action is enabled for a concrete agent, then the action is also enabled for the environment. We write $\mathbb{NB}$ for the class of PIIS with non-blocking environments. We then obtain the following.

Theorem 6. *Let $\mathfrak{M}$ be a P-DAMAS with abstraction $\mathfrak{M}^A$ such that $\mathfrak{M}^A \in \mathbb{NB}$. Then, for every m-indexed formulae ϕ in FO-ECTL$\backslash X$, $M^A(m) \models \phi$ implies $\forall n \geqslant \max(2, m), M(n) \models \phi$.*

The above is the main result of the paper; it outlines a partial procedure to solve the PMCP for P-DAMAS and FO-ECTL$\backslash X$. This takes as input a P-DAMAS $\mathfrak{M}$ and an m-indexed FO-ECTL$\backslash X$ formula ϕ and constructs the abstract P-DAMAS $\mathfrak{M}^A$ as per Definition 12. If the PIIS associated with $\mathfrak{M}^A$ is non-blocking [2], then the abstract DAMAS with up to $\max(2, m)$ agents are checked against the formula. If these satisfy ϕ, then we can conclude that the PMCP is true for $\mathfrak{M}$ and ϕ; otherwise no conclusions can be drawn.

4 Auctions as $\mathbb{AES}$ P-DAMAS

To illustrate the formal machinery and the result in Section 2 and 3, we introduce agent templates for simple English (ascending bid) auctions. We refer to [Easley and Kleinberg, 2010] for a detailed presentation of this type of auctions. First of all, we model the auctioneer and bidders taking part in the auction as the environment and the agent template.

Definition 17 (Auctioneer). *The auctioneer $a = \langle L_a, init_a, Act_a, P_a, \tau_a \rangle$ is such that*

- *L_a is the set of local states defined on set $ctr_a = \{base, t\_out, high\}$ of variables, where t_out is boolean, while $base$ and $high$ range over the rational numbers $\mathbb{Q}$ extended with the "undefined" value* uu.
- *$init_a = \iota_a : ctr_a \to D$, where $\iota_a(base) = \mathrm{uu}$, $\iota_a(t\_out) = \top$, and $\iota_a(high) = \mathrm{uu}$.*
- *Act_a contains guarded commands skip and*

$$\begin{array}{lcll} id_1 & \equiv & t\_out = \bot & \rightsquigarrow \; t\_out := \top \\ id_2 & \equiv & t\_out = \top & \rightsquigarrow \; base := x_2; t\_out := \bot \\ id_3 & \equiv & \top & \rightsquigarrow \; high := x_4 \end{array}$$

  with $id_1 \in GS_a$, $id_2 \in A_a$, and $id_3 \in AE_a$.
- *P_a and τ_a are given as in Def. 4.*

Intuitively, the auctioneer keeps track of the base price $base$ as well as the highest bid $high$ for the auctioned item, and owns a boolean variable t_out to terminate non-deterministically the bidding round. At the start of the bidding process the auctioneer initialises $base$ to a random rational x_2 and t_out to false ($\bot$). Then, she updates the highest bid $high$ and possibly terminate the bidding round. A new round can then be started.

Further, the template for bidders is given as follows.

Definition 18 (Bidder). *The bidder template $t_b = \langle L_b, init_b, Act_b, P_b, \tau_b \rangle$ is such that*

- *$ctr_b = \{tvalue, bid\}$, with both tvalue and bid ranging over $\mathbb{Q} \cup \{\mathrm{uu}\}$.*
- *$init_b = \iota_b : ctr_b \to D$, where $\iota_b(bid) = \mathrm{uu}$ and $\iota_b(tvalue) = \mathrm{uu}$.*
- *Act_b contains guarded commands skip and*

$$\begin{array}{lcl} id_1 & \equiv & \top \rightsquigarrow tvalue := \mathrm{uu}; bid = \mathrm{uu} \\ id_2' & \equiv & (t\_out = \bot) \wedge (tvalue = \mathrm{uu}) \rightsquigarrow tvalue := x_6 \\ id_3 & \equiv & (t\_out = \bot) \wedge (tvalue \neq \mathrm{uu}) \wedge (x_4 \leqslant tvalue) \wedge \\ & & (high \neq \mathrm{uu} \to high < x_4) \wedge \\ & & (bid \neq \mathrm{uu} \to bid < high) \rightsquigarrow bid := x_4 x \end{array}$$

  with $id_1 \in GS_b$, $id_2' \in A_b$, and $id_3 \in AE_b$
- *P_b and τ_b are given as in Def. 4.*

By Def. 18 every bidder template b has a true value $tvalue$, up to which she is happy to bid, as well as current bid. At the beginning she initialises $tvalue$, while bid is set to "undefined". Thereafter, she might choose to bid and then update bid according to the other bidders' offers. At the end of the bidding round, she reinitialises her true value for a new round.

Given the auctioneer and the bidder template as defined above, a P-DAMAS for an English auction is the pair $\mathfrak{M} = \langle a, t_b \rangle$ for the auctioneer a and bidders b. Since base prices, true values, and bids all take rationals as values, $\mathfrak{M}$ is actually an infinite-state system.

On the P-DAMAS $\mathfrak{M}$ we might want to verify properties such as *every agent will eventually win in some execution:* $\phi_{A1} \triangleq \forall^{ag} a \colon EF(win, a)$, where $win(a_i) ::= ((bid, a_i) = high)$. Moreover, we can express that *in at least one execution, every agent bids up to her true value*: $\phi_{A2} \triangleq \forall^{ag} a \colon EF((bid, a) = (t\_value, a))$.

To verify ϕ_{A1} and ϕ_{A2} on $\mathfrak{M}$, we first model check abstraction $\mathfrak{M}^A$ and, if the answer is positive, by Theorems 3 and 6 the result transfers to $\mathfrak{M}$. Notice that this defines a partial verification procedure. If the answer is negative, a possible different abstraction $\mathfrak{M}'^A$ needs to be considered.

5 Conclusions

As argued in the introduction, while data-aware systems have rapidly become common in applications, there is still a lack of techniques capable of providing formal guarantees for systems of agents interacting with these. The difficulty of doing this results both from the possibly infinite amount of data and the unbounded number of agents interacting with it.

In this contribution we addressed these problems and put forward P-DAMAS, a formal model for such systems, then presented a technique for their verification. The key result here is that for the relevant class of P-DAMAS verification is semi-decidable. It should be noted that partial decidability is a common feature in abstraction methodologies, which can normally decide on the truth of a specification in some cases only. Indeed, partial decidability can be useful in several applications of importance, as we showed here in analysing the auction scenario. In future work we plan to extend the present results to yet more expressive languages, including epistemic and strategy logics.

Acknowledgments

The research described in this paper was partly supported by the EPSRC project "Trusted Autonomous Systems"(EP/I00529X) and the French ANR JCJC Project SVeDaS (ANR-16-CE40-0021).

[2] This test can be performed in polynomial time in the size of the agent template and the environment [Kouvaros and Lomuscio, 2013b].

References

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Alur and Henzinger, 1999] R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

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

[Bagheri *et al.*, 2013] B. Bagheri, D. Calvanese, M. Montali, G. Giacomo, and A. Deutsch. Verification of relational data-centric dynamic systems with external services. In *Proceedings of PODS13*, pages 163–174. ACM, 2013.

[Belardinelli and Lomuscio, 2016] F. Belardinelli and A. Lomuscio. Abstraction-based verification of infinite-state reactive modules. In *Proceedings of ECAI16*, pages 725–733, 2016.

[Belardinelli *et al.*, 2012] F. Belardinelli, A. Lomuscio, and F. Patrizi. An abstraction technique for the verification of artifact-centric systems. In *Proceedings of KR12*, pages 319–328, 2012.

[Belardinelli *et al.*, 2014] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of agent-based artifact systems. *Journal of Artificial Intelligence Research*, 51:333–376, 2014.

[Bloem *et al.*, 2015] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. *Decidability of Parameterized Verification*. Morgan and Claypool Publishers, 2015.

[Calvanese *et al.*, 2016] D. Calvanese, M. Montali, F. Patrizi, and M. Stawowy. Plan synthesis for knowledge and action bases. In *Proceedings of IJCAI16*, pages 1022–1029, 2016.

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

[De Masellis *et al.*, 2015] R. De Masellis, D. Lembo, M. Montali, and D. Solomakhin. Semantic enrichment of gsm-based artifact-centric models. *J. Data Semantics*, 4(1):3–27, 2015.

[Deutsch *et al.*, 2009] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proceedings of ICDT09*, pages 252–267. ACM, 2009.

[Deutsch *et al.*, 2016] A. Deutsch, Y. Li, and V. Vianu. Verification of hierarchical artifact systems. In *Proceedings of PODS16*, pages 179–194, 2016.

[Easley and Kleinberg, 2010] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

[Esparza *et al.*, 1999] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proceedings of LICS99*, pages 352–359. IEEE, 1999.

[Fagin *et al.*, 1995] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[Gonzalez *et al.*, 2012] P. Gonzalez, A. Griesmayer, and A. Lomuscio. Verifying GSM-based business artifacts. In *Proceedings of ICWS12*, pages 25–32. IEEE Press, 2012.

[Gonzalez *et al.*, 2015] P. Gonzalez, A. Griesmayer, and A. Lomuscio. Verification of GSM-based artifact-centric systems by predicate abstraction. In *Proceedings of ICSOC15*, volume 9435 of *LNCS*, pages 253–268. Springer, 2015.

[Hamilton, 1978] A. G. Hamilton. *Logic for Mathematicians*. Cambridge University Press, 1978.

[Hoek *et al.*, 2006] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In *Proceedings of AAMAS06*, pages 201–208, 2006.

[Kouvaros and Lomuscio, 2013a] P. Kouvaros and A. Lomuscio. Automatic verification of parametrised interleaved multi-agent systems. In *Proceedings of AAMAS13*, pages 861–868. IFAAMAS, 2013.

[Kouvaros and Lomuscio, 2013b] P. Kouvaros and A. Lomuscio. A cutoff technique for the verification of parameterised interpreted systems with parameterised environments. In *Proceedings of IJCAI13*, pages 2013–2019. AAAI Press, 2013.

[Kouvaros and Lomuscio, 2015a] P. Kouvaros and A. Lomuscio. A counter abstraction technique for the verification of robot swarms. In *Proceedings of AAAI15*, pages 2081–2088. AAAI Press, 2015.

[Kouvaros and Lomuscio, 2015b] P. Kouvaros and A. Lomuscio. Verifying emergent properties of swarms. In *Proceedings of IJCAI15*, pages 1083–1089. AAAI Press, 2015.

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

[Kouvaros and Lomuscio, 2017] P. Kouvaros and A. Lomuscio. Parameterised verification of infinite state multi-agent systems via predicate abstraction. In *Proceedings of AAAI17*, pages 3013–3020. AAAI Press, 2017.

[Lomuscio and Michaliszyn, 2014] A. Lomuscio and J. Michaliszyn. Model checking unbounded artifact-centric systems. In *Proceedings of KR14*, pages 488–497. AAAI Press, 2014.

[Montali and Calvanese, 2016] M. Montali and D. Calvanese. Soundness of data-aware, case-centric processes. *STTT*, 18(5):535–558, 2016.

[Montali *et al.*, 2014] M. Montali, D. Calvanese, and G. De Giacomo. Verification of data-aware commitment-based multiagent system. In *Proceedings of AAMAS14*, pages 157–164. IFAAMAS, 2014.

[Singh and Huhns, 2005] M. Singh and M. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.