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
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 techniques 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 [Berdardinelli 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 \( \text{var} = \{v_0, v_1, \ldots\} \) of variables and a set \( \text{par} = \{x_0, x_1, \ldots\} \) 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 \( \text{cnt}_i \subseteq \text{var} \) of variables. Specifically, \( \{\text{cnt}_t, \text{cnt}_e\} \) forms a partition of \( \text{var} \). Hence, the set \( \text{var} \) can be assumed to be finite. The set \( \text{obs}_i \) of variables that are observable by agent template \( t \) includes all of her controlled variables as well as the variables controlled by the environment: \( \text{obs}_i = \text{cnt}_t \cup \text{cnt}_e \).

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 \( L_i \) of local states, a set \( \text{Act} \) of actions, and a protocol function \( P \). In particular, to introduce a formal account of local state, we consider local interpretations as functions \( \theta_i : \text{cnt}_i \rightarrow D \), i.e., (finite) assignments from the variables in \( \text{cnt}_i \) to values in \( D \). For simplicity, we often identify an interpretation \( \theta_i \) with its range \( \theta_i(\text{cnt}_i) \subseteq D \), whenever domain \( \text{cnt}_i \) is clear by the context. Then, a local state \( l \in L_i \) of agent template \( t \) includes all values of its observed variables in \( \text{obs}_i \), i.e., \( l = \theta_i \cup \theta_e \). Since the domain \( D \) is infinite in general, the set \( L_i \) of local states is also infinite.

To define the individual actions in \( \text{Act} \) and the protocol \( P \), we introduce a first-order language built on variables, parameters and relation symbols \( = \) and \( \leq \).

Definition 1 (FO-formulas). First-order formulas are defined according to the following BNF, where \( z, z' \in \text{var} \cup \text{par} \) and \( x \in \text{par} \):

\[
\phi ::= z = z' \mid z \leq z' \mid \neg \phi \mid \phi \land \phi \mid \forall x \phi
\]

The symbols \( = \), \( \leq \), \( \top \), \( \bot \), connectives \( \land \), \( \lor \), 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 \( \gamma \) over \( \text{var} \) and \( \text{par} \) is an expression

\[
\text{id} \equiv g(x_1, \ldots, x_k) \sim v_1 := x_1; \ldots; v_k := x_k
\]

where (i) \( \text{id} \) 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 \( \sigma : \text{par} \rightarrow D \) of parameters, then the command \( \gamma \) is enabled for execution. By executing \( \gamma \) we set each variable \( v_i \) to value \( \sigma(x_i) \in D \). In particular, the skip command can be represented as \( \top \sim \epsilon \), where \( \epsilon \) is the empty sequence. We say that \( v_1, \ldots, v_k \) are the variables controlled by \( \gamma \), and denote this set by \( \text{cnt}(\gamma) \), while the variables in \( g \) are the observable variables \( \text{obs}(\gamma) \) [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 \( \phi \) is given meaning by a finite interpretation \( \sigma : \text{fr}(\phi) \rightarrow D \) that assigns values in \( D \) to the free parameters in \( \phi \). A reinterpretation \( \sigma^v \) coincides with \( \sigma \), but assigns value \( v \in D \) to parameter \( x \in \text{fr}(\phi) \). Given \( z \in \text{var} \cup \text{par} \), \( (\theta, \sigma)(z) = \theta(z) \) for \( z \in \text{var} \), and \( (\theta, \sigma)(z) = \sigma(z) \) for \( z \in \text{par} \).

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

\[
(\theta, \sigma) \models z = z' \quad \text{iff} \quad (\theta, \sigma)(z) = (\theta, \sigma)(z')
\]

\[
(\theta, \sigma) \models z \leq z' \quad \text{iff} \quad (\theta, \sigma)(z) \leq (\theta, \sigma)(z')
\]

\[
(\theta, \sigma) \models \forall x \phi \quad \text{iff} \quad \text{for all } u \in \theta(\text{var}), \sigma^u \models \phi
\]

The interpretation of FO-formulas is completely standard, but for quantification that takes values from the finite image \( \theta(\text{var}) = \{u \in D \mid u = \theta(v) \text{ for some } v \in \text{var}\} \) of \( \text{var} \). 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 \( \theta(\text{var}) \) is finite.

Definition 4 (Agent template). The agent template is a tuple

\[
t = (L, \text{init}, \text{Act}, P, \tau)
\]

where

- \( L = \{\theta_t \cup \theta_e \mid \theta_i : \text{cnt}_i \rightarrow D \mid i \in \{t, e\}\} \)
- \( \text{init} = I_t \cup I_e \), where \( I_t : \text{cnt}_t \rightarrow D \) and \( I_e : \text{cnt}_e \rightarrow D \)
- \( P : L \rightarrow \varphi(\text{Act}) \setminus \{\emptyset\} \)
- \( \tau : L \times \text{Act} \rightarrow L \)

is such that, for every \( L \in P(\{\alpha \in \text{Act} \mid (l, \sigma) \models \gamma\};

\[
\tau : L \times \text{Act} \rightarrow L \text{ is such that (i) } \tau(l, \alpha, \alpha_e) \text{ is defined only if } \alpha \in P(l); \text{ and (ii) } \tau(l, \alpha, \alpha_e) = l' \text{ iff for every variable } v_j \in \text{cnt}(\gamma) \text{ and } w_e \in \text{cnt}(\gamma_e), \theta_f(v_j) = \alpha_g(x_j) \text{ and } \theta_e(w_j) = \alpha_e(y_j); \text{ while all other variables do not change value.}
\]

The environment is similarly defined. Observe, however, that its set of local states is defined only on \( \theta_e \) and its tran-
sition function is defined only on its current state \( l_e \) and action \( \alpha_e \). The actions of the agent template are partitioned as \( \text{Act} = A \cup AE \cup GS \), 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 \( id(X) = \{ id_e \mid (\gamma, \sigma) \in X \} \) 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 \( id(GS_e) = id(GS_t) \).

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 \( \mathfrak{M} = (t, e) \), 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 \( Ag(n) = \{ t_j \mid 1 \leq j \leq n \} \) 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 \( \text{var}(n) = \{ v \times \{ 1, \ldots, n \} \mid v \in ctri \} \cup ctrie \) of variables, where agent \( t_j \) controls the variables in \( ctri_j = \{ v_j \in \text{var}(n) \mid v \in ctri \} \) and observes the variables in \( \text{obs}_j = ctrie_j \cup ctrie \). This is consistent with the requirement that \( \{ ctri_1, \ldots, ctri_n, ctrie \} \) form a partition of \( \text{var}(n) \).

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

- \( L_j = \{ \gamma \cup \theta_e \mid \theta_i : ctri_j \rightarrow D \text{ for } i \in \{ j, e \} \} \);
- \( \text{init}_j = \{ t_j \cup \theta_e \mid t_j : ctri_j \rightarrow D \text{ is such that } t_j(v_j) = x \iff t_k(v) = x \} \);
- \( \text{Act}_j = \{ (\gamma', \sigma) \mid (\gamma, \sigma) \in \text{Act}_t \} \), where \( \gamma' \) is obtained from \( \gamma \) by replacing every variable \( v \in ctri(\gamma) \) by \( v_j \);
- The protocol \( P_j \) and the transition function \( \tau_j \) 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 \( \theta_i : ctri_j \rightarrow D \) is an interpretation for the \( j \)-th instantiation of template \( t \). Equivalently, global states can be represented as functions \( s : \text{var}(n) \rightarrow D \), i.e., finite interpretations of the variables in \( \text{var}(n) \) with values in \( D \) such that for every \( v_j \in \text{var}(n) \), \( s(v) = \theta_j(v) \). As anticipated above, any state \( s \) is well-defined as \( \text{var}(n) \) 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 \( \text{Act} = \prod_{ag \in \text{Ag}(n) \cup \{ e \}} \text{Act}_{ag} \) be the set of joint actions. For \( \vec{s} \in \text{Act} \), consider \( \vec{s}_{ag} \) 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 \( \tau : \mathcal{G} \times \text{Act} \rightarrow \mathcal{G} \) is defined as follows:

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

Above \( \tau \) 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 \( \tau_{ag} \), we have that for every \( ag \in Ag(n) \) \( \cup \{ e \} \), \( \vec{s}_{ag} \) is \( P_{ag}(s_{ag}) \).

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 \( M(n) = \langle S, \text{init}, \tau \rangle \), where:

- \( \text{init} = \prod_{ag \in \text{Ag}(n) \cup \{ e \}} \text{init}_{ag} \); \( \tau \) is the global transition function (Definition 7); \( S \) is the closure of init according to \( \tau \).

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 \( \tau \). 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\(\perp X \) (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\(\perp X \)).** Indexed first-order formulas over agent parameters \( apar \) are defined according to the following BNF, where \( z = \langle v, a \rangle \)
Parameterised model checking involves checking an unbounded number of systems. Since P-DAMAS extend broadcast protocols whose PMCP is undecidable [Epsarza 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\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\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 \text{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 \text{cnt}_i to D\A.

Further, given P-DAMAS \mathcal{M} = \langle t, e\rangle, the abstract P-DAMAS \mathcal{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 \text{M}(n). Hence, M\A(n) is a submodel of M(n). In particular, M\A(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 $\pi$ 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 $\phi$ in FO-ECTL\(\times\), 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 $M^A$ we can associate a PIIS $M^PA$ whose concrete systems satisfy the same FO-ECTL\(\times\) formulas as the equally populated concrete systems from $M^A$. 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., $obs_e = cnt_i$; (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 $M^PA$. 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 $id \equiv g(x_1, \ldots , x_k) \leadsto v_1 := x_1; \ldots ; v_k := x_k$ and $id' \equiv g'(x'_1, \ldots , x'_k) \leadsto v'_1 := x'_1; \ldots ; v'_k := x'_k$, is defined as the guarded command $id \otimes id' \equiv g(x_1, \ldots , x_k) \land g'(x'_1, \ldots , x'_k) \leadsto v_1 := x_1; \ldots ; v_k := x_k; v'_1 := x'_1; \ldots ; v'_k := x'_k$.

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 $\gamma$ be an agent-environment command $id \equiv g(x_1, \ldots , x_k) \leadsto v_1 := x_1; \ldots ; v_k := x_k$. The AE initiator command of $\gamma$, $\gamma[1]$, is the agent-environment command $id[1] \equiv g(x_1, \ldots , x_k) \land ae_{sync} = \bot \leadsto ae_{sync} = T$. The AE broadcast command $\gamma[\bot]$ of $\gamma$ is the global-synchronous command $id[\bot] \equiv g(x_1, \ldots , x_k) \land ae_{sync} = \bot \leadsto v_1 := x_1; \ldots ; v_k := x_k; ae_{sync} = \bot$.

$AE$-synchronisation commands enable the PIIS agents to simulate the agent-environment transitions of the DAMAS agents. In particular the $AE$ initiator command $\gamma[1]$ 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 $\gamma[\bot]$ by setting the (fresh) boolean variable $ae_{sync}$ to $T$. 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 $M^PA$ associated with $M^A$.

**Definition 15 (Associated PIIS).** The PIIS $M^PA = \langle t^A, e^A \rangle$ associated with P-DAMAS $M^A = \langle t^A, e^A \rangle$ 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 $e^PA$:

- $Act_{t^PA}$: $A \subseteq D$ is the set of asynchronous actions; $\{a[?] | a \in AE_e\}$ is the set of agent-environment actions; and $AE_e \cup \{a[!] \times a_1[!] | a \in AE_t, a_1 \in AE_e, id_a = id_{a_1}\} \cup \{a \times a_1 | a \in GS_t, a_1 \in GS_e, id_a = id_{a_1}\}$ is the set of global-synchronous actions.

- $Act_{e^PA}$: $\{a[?] | a \in AE_e\}$ is the set of agent-environment actions and $GS_e \cup \{a[!] | a \in GS_e\}$ 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 $a \in \{1, \ldots , n\}$, $s(\langle ae_{sync}, ag \rangle) = q(\langle ae_{sync}, ag \rangle) = \bot$.

Following the above definition we show that related states satisfy the same FO-ECTL\(\times\) formulas. Since the initial states of corresponding concrete systems are related, the systems satisfy the same FO-ECTL\(\times\) 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 $q \xrightarrow{\alpha''} q''$ and $s' \approx q''$, where $\alpha'$ is defined by $\alpha'.j = \text{skip for } j \neq i \neq e$, $\alpha'.i = \alpha.i[?]$, and $\alpha'.e = \alpha.e[?];$ $\alpha''$ 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 $\alpha'$ 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 $M$ be a P-DAMAS with abstraction $M^A$. Let $M^PA(n)$ be the PIIS associated with $M^A$. Then, for every formula $\phi$ in FO-ECTL\(\times\), $M^A(n) \models \phi$ 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, 2013]. 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 \( \mathbb{N} \) for the class of PIIS with
non-blocking environments. We then obtain the following.

**Theorem 6.** Let \( \mathcal{M} \) be a P-DAMAS with abstraction \( \mathcal{M}^A \)
such that \( \mathcal{M}^A \in \mathbb{N} \). Then, for every \( m \)-indexed for-

\[ X \] 

mulae \( \phi \) in FO-ECTL, \( M^A(m) \models \phi \) implies \( \forall n \geq \max(2,m), M(n) \models \phi \).

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

4 Auctons as AES P-DAMAS

To illustrate the formal machinery and the result in Section 2
and 3, we introduce agent templates for simple English (as-
cending 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 \( a = \langle L_a, \text{init}_a, \text{Act}_a, P_a, \tau_a \rangle \) is such that

- \( L_a \) is the set of local states defined on set \( \text{ctr}_a = \{\text{base}, \text{t}_\text{out}, \text{high}\} \) of variables, where \( \text{t}_\text{out} \) is boolean, while base and high range over the rational
numbers \( \mathbb{Q} \) extended with the “undefined” value \( uu \).
- \( \text{init}_a = \text{i}_a : \text{ctr}_a \rightarrow D \), where \( \text{i}_a(\text{base}) = uu \) and \( \text{i}_a(\text{t}_\text{out}) = T \) and \( \text{i}_a(\text{high}) = uu \).
- \( \text{Act}_a \) contains guarded commands skip and

\[ id_1 \equiv \text{t}_\text{out} = \bot \wedge T \rightarrow \text{t}_\text{out} = T; \]

\[ id_2 \equiv \text{t}_\text{out} = T \wedge \text{base} := x_2; \text{t}_\text{out} := \bot \]

\[ id_3 \equiv T \rightarrow \text{high} := x_4 \]

with \( id_1 \in G_{\text{Sa}}, id_2 \in A_a \) and \( id_3 \in A_{E_a} \).
- \( P_a \) and \( \tau_a \) are given as in Def. 4.

Intuitively, the auctioneer tracks the base of the price
base as well as the highest bid \( \text{high} \) for the auctioned
item, and owns a boolean variable \( \text{t}_\text{out} \) to terminate non-
deterministically the bidding round. At the start of the bidding
process the auctioneer initialises \( \text{base} \) to a random rational
\( x_2 \) and \( \text{t}_\text{out} \) to false (\( \bot \)). Then, she updates the highest
bid \( \text{high} \) 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 \( b = \langle L_b, \text{init}_b, \text{Act}_b, P_b, \tau_b \rangle \) is such that

- \( \text{ctr}_b = \{\text{t}_\text{value}, \text{bid}\} \), with both \( \text{t}_\text{value} \) and \( \text{bid} \) ranging
over \( \mathbb{Q} \cup \{uu\} \).

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

- \( \text{init}_b = \text{i}_b : \text{ctr}_b \rightarrow D \), where \( \text{i}_b(\text{bid}) = uu \) and \( \text{i}_b(\text{t}_\text{value}) = uu \).
- \( \text{Act}_b \) contains guarded commands skip and

\[ id_1 \equiv \tau \rightarrow \text{t}_\text{value} := uu; \text{bid} = uu \]

\[ id_2 \equiv (\text{t}_\text{out} = \bot) \wedge (\text{t}_\text{value} = uu) \wedge \text{t}_\text{value} := x_6 \]

\[ id_3 \equiv (\text{t}_\text{out} = \bot) \wedge (\text{t}_\text{value} = uu) \wedge (x_4 \leq \text{t}_\text{value}) \wedge \]

\( \text{high} \neq uu \rightarrow \text{high} < x_4 \wedge \)

\( \text{bid} \neq uu \rightarrow \text{bid} < \text{high} \) \wedge \text{bid} := x_4 \]

with \( id_1 \in G_{S_b}, id_2 \in A_b \) and \( id_3 \in A_{E_b} \).

By Def. 18 every bidder template \( b \) has a true value \( \text{t}_\text{value} \),
up to which she is happy to bid, as well as current \( \text{bid} \). At the
beginning she initialises \( \text{t}_\text{value} \), while \( \text{bid} \) is set to “unde-
defined”. Thereafter, she might choose to bid and then update
\( \text{bid} \) 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 \( \mathcal{M} = \langle a, b \rangle \) for the auctioneer \( a \) and bidders \( b \). Since base prices,
true values, and bids all take rationals as values, \( \mathcal{M} \) is actually
an infinite-state system.

On the P-DAMAS \( \mathcal{M} \) we might want to verify proper-
ties such as every agent will eventually win in some exe-
cution: \( \phi_{A1} \equiv \forall a : EF(\text{win}(a)) \), where \( \text{win}(a) := (\text{bid}, a_1) = (\text{high}) \). Moreover, we can express that in at
least one execution, every agent bids up to her true value:
\( \phi_{A2} \equiv \forall a : EF((\text{bid}, a) = (\text{t}_\text{value}, a)) \).

To verify \( \phi_{A1} \) and \( \phi_{A2} \) on \( \mathcal{M} \), we first model check abstraction \( \mathcal{M}^A \) and, if the answer is positive, by Theorems 3 and 6
the result transfers to \( \mathcal{M} \). Notice that this defines a partial
verification procedure. If the answer is negative, a possible
different abstraction \( \mathcal{M}^A \) 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 sys-
tems 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 ap-
lications 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 sup-
ported by the EPSRC project “Trusted Autonomous Sys-
tems” (EP/I00529X) and the French ANR JCJC Project
SVeDaS (ANR-16-CE40-0021).
References


