Finite Abstractions for the Verification of Epistemic Properties in Open Multi-Agent Systems

Francesco Belardinelli  
Laboratoire IBISC  
Université d’Evry, France  
belardinelli@ibisc.fr

Davide Grossi  
Department of Computer Science  
University of Liverpool, UK  
d.grossi@liverpool.ac.uk

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

Abstract

We develop a methodology to model and verify open multi-agent systems (OMAS), where agents may join in or leave at run time. Further, we specify properties of interest on OMAS in a variant of first-order temporal-epistemic logic, whose characterising features include epistemic modalities indexed to individual terms, interpreted on agents appearing at a given state. This formalism notably allows to express group knowledge dynamically. We study the verification problem of these systems and show that, under specific conditions, finite bisimilar abstractions can be obtained.

1 Introduction

Modal temporal-epistemic logic has long been adopted as a formalism for reasoning about multi-agent systems (MAS). In its basic setting it consists of either the linear or the branching version of discrete-time temporal logic, augmented with knowledge modalities for the agents in the system. Several properties of MAS (e.g., perfect recall, no learning, synchronicity) have been axiomatised on the widely adopted semantics of Interpreted Systems [Fagin et al., 1995]. In the past decade several model checking methodologies and toolkits that use temporal-epistemic specification languages have been developed [Penczek and Lomuscio, 2003; Gammie and van der Meyden, 2004; Lomuscio et al., 2009].

Two key assumptions are made in the basic setting of the formalism above. Firstly, facts are expressed in propositional terms. Secondly, the number of agents is finite and given at design time. As a consequence, the indexes of individual knowledge operators are constants in a finite set $Ag$ of agents, while the indexes for group knowledge operators are finite subsets of $Ag$. Propositions have been made to overcome the first limitation by introducing first-order versions of temporal-epistemic logic both on quantified versions of interpreted systems [Belardinelli and Lomuscio, 2012] and on artifact-centric multi-agent systems [Belardinelli et al., 2014]. These approaches surmount the shortcomings of a purely propositional language by extending the syntax to first-order formulas. In some cases completeness can be retained [Belardinelli and Lomuscio, 2012] and verification can in principle be performed on finite abstractions [Belardinelli and Lomuscio, 2013; Belardinelli et al., 2014].

Regarding the second limitation, proposals have been put forward to consider a set of objects that vary at design time: the set of agents is normally considered to be finite in each system run. This is a sensible assumption in many scenarios, but there are applications of MAS (e.g., e-commerce, smart grids) where an unbounded number of agents may freely enter and leave the system at run time. There is, therefore, a need to account for the unbounded and possibly infinite agents joining in or leaving an open MAS. In this setting it is still of interest to reason about their evolution and what they know individually and collectively. For example, in an auction setting, such as fishmarkets [Rodríguez-Aguilar et al., 1998], different agents attend different auctions at run time. Nonetheless, all of them, however many they may be, will eventually know what the reserve price for a particular good is. Formally, such a temporal-epistemic specification can be expressed in the proposed formalism as:

$$\forall x (\text{Good}(x) \rightarrow \exists y \, \forall z \, K_z \text{Price}(y, x))$$  \hspace{1cm} (1)

(1) intuitively expresses that any good $x$ has always a price $y$ that will eventually be learnt by all agents $z$ currently attending the auction. A key feature of this specification is that agents appear as quantifiable terms in the logical language and appear as such in the indexes of the epistemic operators. In particular, compare the subformula $\forall z K_z \text{Price}(y, x)$ of (1), where the quantification domain of $\forall z$ changes depending on the state, with the standard temporal-epistemic formula $\bigwedge_{a \in Ag} K_a \text{Price}(y, x)$, which assumes $Ag$ fixed. In this paper we propose a formalism accounting for (1): we show that, while the verification problem is undecidable in general, bounded MAS admit finite abstractions.

Related Work. A quantified doxastic logic, with modalities indexed by variables, was introduced in [Lomuscio and Colombetti, 1996]. However, this focused on a 3-valued semantics in view of providing sound axiomatisations. In contrast, here we consider a 2-valued semantics and the model checking problem. Various classes of models and types of quantifications have been developed recently to account for Artifact-centric Systems [Belardinelli et al., 2012; 2014; Hariri et al., 2013]. While our work is influenced by the techniques introduced therein, none of these contributions deals with open MAS with a possibly infinite number of agents.
In [Montali et al., 2014] agents appear in the relational structure, but there is no explicit quantification on them. This feature, however, may be useful in a number of scenarios, as we argue in this paper. There are also similarities with recent work on parametrised verification of MAS [Kouvaros and Lomuscio, 2013a; 2013b]. However, parametrised verification aims at establishing whether properties hold irrespective of the finite but unbounded number of agents in the system; here we deal with an infinite set of agents which we only bound at a state and not in the whole model. Closer to our approach is [Belardinelli and Grossi, 2015] which introduces a semantics for dynamic agent networks; however, epistemic operators are not discussed there.

**Scheme of the paper.** In Section 2 we formalise open multi-agent systems in our setting and introduce a novel first-order temporal-epistemic logic CTLK. We state the model checking problem for this setting (Section 2.2) and illustrate the formal machinery with a use case (Section 2.3). Section 3 contains the main theoretical results on the existence of finite, bisimilar abstractions. We conclude and point to future work in Section 4.

## 2 Open Multi-agent Systems

In this section we present a formalism to reason about open multi-agent systems (OMAS). A key feature of OMAS is that agents may join and leave the system at run time. We then put forward a first-order version of the temporal-epistemic logic CTLK to reason about OMAS, that allows us to index knowledge operators with variables. We conclude by formulating the model checking problem for OMAS. Since we wish to account for possibly infinite domains of objects and agents we import some basic terminology from related literature [Abiteboul et al., 1995; Belardinelli et al., 2014].

**Definition 1 (Database schema and instance)** A database schema is a finite set \( D = \{ P_1/q_1, \ldots, P_n/q_n \} \) of predicate symbols \( P \) with arity \( q \in \mathbb{N} \).

Given a (possibly infinite) interpretation domain \( X \), a \( D \)-instance over \( X \) is a mapping \( D \) associating each predicate symbol \( P \) to a finite \( q \)-ary relation on \( X \), i.e., \( D(P) \subseteq X^q \).

For a database schema \( D \), \( D(X) \) is the set of all \( D \)-instances on \( X \); while the active domain \( \text{dom}(D) \) is the finite set \( \bigcup_{P \in D} \{ u_1, \ldots, u_q \in X \mid \langle u_1, \ldots, u_q \rangle \in D(P) \} \) of all individuals occurring in some predicate interpretation \( D(P) \). Further, the *primed version* of a database schema \( D \) as above is the schema \( D' = \{ P'_1/q_1, \ldots, P'_n/q_n \} \). Then, the disjoint union \( D \uplus D' \) of \( D \)-instances \( D \) and \( D' \) is the \( (D \uplus D') \)-instance s.t. (i) \( D \uplus D'(P) = D(P) \), and (ii) \( D \uplus D'(P') = D'(P') \). Hereafter, primed versions and disjoint unions are used to account for the temporal evolution of a database from the previous state \( D \) to the next state \( D' \).

### 2.1 Agents in OMAS

To introduce OMAS, we import some preliminary notions from [Belardinelli and Grossi, 2015]. Hereafter we assume a finite number of agent types \( T_0, \ldots, T_k \). Each agent type \( T \) comprises (i) a local database schema \( D_T \), and (ii) a finite set \( \text{Act}_T \) of parametric actions \( \alpha(X) \). Hence, agents of the same type share the database schema and available actions. For every agent type \( T \), \( \text{Ag}_T \), \( \text{Ag}_{T'} \), \ldots are (possibly infinite) sets of agent names. In the rest of the paper, the interpretation domain \( X \) contains a set \( \text{Ag}_T \) of agent names for each type \( T \), i.e., \( X = \text{Ag}_T \cup U \) for \( \text{Ag}_T = \bigcup_{T'} \text{Ag}_T \) and some other (possibly empty) set \( U \) of elements. We will also consider a set \( \text{Con} \subseteq X \) of constants, including names for agents. To describe the temporal evolution of OMAS, we define protocols for agent types. To do so, we first introduce isomorphisms on database instances.

**Definition 2 (Instance Isomorphism)** Instances \( D \in D(X) \) and \( D' \in D(X') \) are isomorphic, or \( D \simeq D' \), iff for some bijection \( \iota : \text{dom}(D) \cup \text{Con} \leftrightarrow (\text{dom}(D') \cup \text{Con}) \) that is the identity on \( \text{Con} \); (ii) \( \iota \) is type-preserving, i.e., for every type \( T \), \( \iota \) is a bijection from \( \text{dom}(D) \cup \text{Con} \) to \( \text{dom}(D') \cup \text{Con} \). Whenever the above holds, we say that \( \iota \) is a witness for \( D \simeq D' \) and write \( D \simeq D' \) to state this explicitly. While isomorphisms depend on the set \( \text{Con} \) of constants, in what follows we consider \( \text{Con} \) fixed and omit it.

We now introduce the local protocol \( Pr_T \) for a type \( T \).

**Definition 3 (Protocols)** Given domain \( X \), \( Pr_T \) is a function from \( D_T(X) \) to \( 2^{\text{Act}_T(X)} \), where \( \text{Act}_T(X) \) is the set of ground actions \( \alpha(\vec{u}) \), for \( \alpha(\vec{x}) \in \text{Act}_T \) and \( \vec{u} \in X^q \).

By Def. 3 the protocol \( Pr_T \) returns a ground action in \( \text{Act}_T(X) \) for every \( D_T \)-instance. In the rest of the paper we assume the following requirement on protocols:

\[ \text{for all instances } D, D' \subseteq D_T \text{, if } D \simeq D' \text{ then } \alpha(\vec{u}) \in Pr_T(D) \text{ iff } \alpha(\vec{u}) \in Pr_T(D') \quad (*) \]

So, by requirement (**) isomorphic states allow “isomorphic” ground actions. Most OMAS of interest satisfy (**). For example, in an English auction an agent may make a valid bid as long as the bid is above the current best price.

We now introduce the notion of agent.

**Definition 4 (Agents)** Given an agent name \( a \in \text{Ag}_T \) of type \( T \), an agent is a tuple \( a = (D_T, \text{Act}_T, Pr_T) \) where \( D_T \), \( \text{Act}_T \), and \( Pr_T \) are defined as above.

We assume a finite number of agent types, but we do not assume a bound on the number of agents of each type in any concrete instantiation of the system. This is common place in OMAS, such as in services, auctions, etc., whereby engineers have prior knowledge of the behaviour of the agent types without knowing how many instances of each type will be executed at runtime. We provide an example of this in Section 2.3. Agents as in Def. 4 are related to the notion of agent templates introduced in [Kouvaros and Lomuscio, 2013a; 2013b]. However, while the latter assumes that any concrete run admits a finite number of agents built on these types, we do not make this assumption here.

In the following an agent is often identified with her name; therefore we write \( a = (D_a, \text{Act}_a, Pr_a) \) and omit the type. By Def. 4 a local state \( l \in D_a(U \cup \text{Ag}_T) \) encodes the knowledge of agent \( a \) about the elements in \( U \) as well as fellow agents in \( \text{Ag}_T \). Thus, a fundamental difference with the standard approach to multi-agent systems [Parikh and Ramanujam, 1985;]
Fagin et al., 1995; Wooldridge, 2001] is that the agent’s information is structured as a relational database.

We can now introduce OMAS to represent the interactions amongst agents, beginning with the notion of global state.

**Definition 5 (Global States)** Given a finite subset $A \subseteq Ag$ of agents $a_i = (D_i, Act_i, Pr_i)$ defined on domain $X = U \cup Ag$, for $i \leq n$, a global state is a tuple $s = (l_0, \ldots, l_n)$ of instances $l_i \in D_i(X)$ s.t. $\bigcup_{i \leq n} \text{dom}(l_i) \cap Ag \subseteq A$.

Note that, while we admit an infinite number of agents in existence, only a finite number of them can be active at any given time, and different agents can be active at different times, thus accounting for the openness of the system. Also by Def. 5, a global state $s$ comprises at least all agents appearing in its active domain $\text{dom}(s) = \bigcup_{i \leq n} \text{dom}(l_i)$. For instance, if agent $a$ appears in the local state $l_b \in D_b(X)$ of agent $b \in A$, and thus $a \in \text{dom}(s)$, then $a$ also belongs to $A$. By assuming a fixed enumeration of agents, we will identify global states containing the same local states for the same agents, possibly in a different order. Further, let $ag$ be the function that for any global state $s = (l_0, \ldots, l_n)$ returns the set $ag(s) = \{a_0, \ldots, a_n\}$ of agents s.t. $l_i \in D_a(X)$ for $i \leq n$. By the requirement above on global states, for every state $s$, $ag(s) \cap Ag \subseteq ag(s)$. We let $G$ be the set $\bigcup_{n \in \mathbb{N}} \prod_{i \leq n} D_i(X)$ of all global states. As a consequence, $G$ is infinite whenever $X$ is.

To account for the knowledge of agents, we say that states $s = (l_0, \ldots, l_n)$ and $s' = (l'_0, \ldots, l'_m)$, of possibly different lengths, are epistemically indistinguishable for agent $a_i$, or $s \sim_i s'$, iff $a_i \in ag(s)$, $a_i \in ag(s')$, and $l_i = l'_i$. Since $s$ and $s'$ can be tuples of different length, an agent does not generally know the exact number of active agents at each moment, nor their identity. Observe that if $\emptyset \notin ag(s)$, then the set $\{s' \in G \mid s' \sim_a s\}$ is empty. That is, if agent $a$ is not active in state $s$, then no state is indistinguishable for her. We elaborate more on this point in Section 2.2.

Finally, we introduce open multi-agent systems.

**Definition 6 (OMAS)** Given a (possibly infinite) domain $X = Ag \cup U$ containing a (possibly infinite) set $Ag = \{a_0, a_1, \ldots\}$ of agents $a_i = (D_i, Act_i, Pr_i)$, an open multi-agent system is a tuple $\mathcal{P} = (Ag, U, I, \tau)$ where

- $I$ is the set of initial states $s_0$ for some finite $ag(s_0) \subseteq Ag$;
- $\tau : G \times Act(X) \rightarrow 2^G$ is the global transition function, where $Act$ is the set of joint (parametric) actions, and $\tau((l_0, \ldots, l_n), (\alpha_0(l_0), \ldots, \alpha_n(l_n)))$ is defined iff $\alpha_i(l_i) \in Pr_i(l_i)$ for every $i \leq n$.

An OMAS describes all system’s executions from an initial state $s_0 \in I$, according to the global transition function $\tau$, which returns the successor states $\tau(s, \alpha)$ for all agents in $s$. Since the domain $X$ is typically infinite, OMAS are infinite-state systems in general. Specifically, OMAS are open and dynamic as global states may be tuples of different length, comprising different agents. Differently from most literature on MAS [Parikh and Ramanujam, 1985; Fagin et al., 1995; Wooldridge, 2001], which assumes that the set of agents is finite and fully specified at design time, here the successor states returned by the transition function may contain fewer or more agents w.r.t. the current state.

We now state a requirement on joint actions in OMAS. To introduce it, we first extend isomorphisms to global states.

**Definition 7 (State Isomorphism)** The global states $s \in G$ and $s' \in G'$ are isomorphic, or $s \simeq s'$, iff for some bijection $\iota : \text{dom}(s) \cup \text{Con} \cup ag(s) \rightarrow \text{dom}(s') \cup \text{Con} \cup ag(s')$, for every $a_i \in ag(s)$, $\iota$ is a witness for $l_i = \iota(l_i)$.

Any function $\iota$ as above is a witness for $s \simeq s'$, also indicated as $s \not\simeq s'$. As for instance isomorphisms, $\simeq$ is an equivalence relation, and by Def. 7 isomorphic states are tuples of the same length. In the rest of the paper we impose the following requirement on the transition functions in OMAS:

- For all states $s, s' \in G$, $s \not\simeq s'$ implies that $t \in \tau(s, \alpha) \iff t \in \tau(s', \alpha)$.

Similarly to protocols, requirement (+) guarantees that actions performed with “isomorphic” values in isomorphic states, also return isomorphic states. In Section 2.3 we will discuss an OMAS satisfying (+); but similar assumptions are common place in database theory and the theory of programming languages [Harri et al., 2013; Deutsch et al., 2007].

We now introduce some useful notation. We define the transition relation $s \rightarrow s'$ on global states iff $s \overset{\alpha}{\rightarrow} s'$ for some joint ground action $\alpha$, i.e., $s' \in \tau(s, \alpha)$. An $s$-run $r$ is an infinite sequence $s_0 \rightarrow s_1 \rightarrow \cdots$, with $s_0 = s$. For $n \in \mathbb{N}$, we set $r(n) = s_n$. A state $s'$ is reachable from $s$ iff $s' = r(i)$ for some $s$-run $r$ and $i \geq 0$. Hereafter we enforce seriality on the transition relation $\rightarrow$ by assuming skip actions. Further, we introduce $S$ as the set of states reachable from some initial state $s_0 \in I$. Since the domain $X$ may be infinite, the set $S$ of reachable states is also infinite in principle. Indeed, OMAS are infinite-state systems in general. Finally, we will refer to the global database schema $D_s = D_0 \cup \cdots \cup D_n$ of a state $s = (l_0, \ldots, l_n)$, and the corresponding $D_s$-instance $D_s(P) = \bigcup_{i \leq n} l_i(P)$, for $P \in D_s$. Therefore, we suppose that each agent has a truthful, yet limited, view of the global database $D_s$. Also, the disjoint union $s + s'$ is defined as state $s'' = (l''_0, \ldots, l''_m)$ on $ag(s) \cup ag(s')$ s.t. (i) if $a_i \in ag(s) \cap ag(s')$ then $l''_i = l_i \cup l'_i$; (ii) if $a_i \in ag(s) \backslash ag(s')$ then $l''_i = l_i$; and (iii) if $a_i \in ag(s') \backslash ag(s)$ then $l''_i = l'_i$.

2.2 The Specification Language FO-CTLK

We now introduce FO-CTLK, a first-order extension of the temporal epistemic logic CTLK, as a specification language for OMAS. Differently from other quantified temporal-epistemic logics [Belardinelli et al., 2014], FO-CTLK features an expressive formulation of the epistemic operators that can be indexed by individual terms. Below we consider a set $\text{Var}$ of individual variables containing a set $\text{Var}_{Ag}$ of variables for agents, as well as the database schema $D = \bigcup_{\text{type} \in T} D_T$. Terms $t, t', \ldots$ are either variables or constants in $\text{Con}$.

**Definition 8 (FO-CTLK)** The $\text{FO-CTLK}_x$ formulas are defined in RNF as follows:

- $\varphi ::= Pr(\bar{t}) \mid t = t' \mid \varphi \land \varphi \mid \forall x \varphi \mid AX \varphi \mid A_x U \varphi \mid E_x U \varphi \mid K_a \varphi \mid K_x \varphi$
where \( t, t' \) are terms, \( P \in D, a \in \text{Con} \cap \text{Ag}, z \in \text{Var}_{\text{Ag}}, \) and \( \bar{t} \) is a \( q \)-tuple of terms.

The temporal formulas \( AX \varphi \) and \( A \varphi U \varphi' \) (resp. \( E \varphi U \varphi' \)) are read as “for all runs, next \( \varphi' \)” and “for every (resp. some) run, \( \varphi \) until \( \varphi' \)”. The epistemic formula \( K_t \varphi \) means that “the agent denoted by \( t \) knows \( \varphi' \)”. The fact that epistemic modalities are indexed to terms represents a significant difference w.r.t. standard approaches. Free and bound variables are defined as standard, as well as sets \( \text{var}(\phi) \) (resp. \( fr(\phi), \text{con}(\phi) \)) of all variables (resp. free variables, constants) in \( \phi \). Notice that \( z \in fr(K_z \varphi) \) and \( \alpha \in \text{con}(K_\alpha \varphi) \). The same symbols are sometimes used to refer to individual variables and action parameters, the context will disambiguate.

To define the satisfaction of an FO-CTLK_\( \pi \) formula on an OMAS, we introduce the notion of an assignment \( \sigma : \text{Var} \rightarrow X \) s.t. for every \( z \in \text{Var}_{\text{Ag}}, \sigma(z) \in A_0 \). We denote by \( \sigma_a^x \) the assignment s.t. \( (i) \sigma_a^x(x) = u; \) and \( (ii) \sigma_a^x(x') = \sigma(x') \) for every \( x' \) different from \( x \). Also, \( \sigma(c) = c \) for all \( c \in \text{Con} \).

**Definition 9 (Semantics of FO-CTLK_\( \pi \))** We define whether an OMAS \( \mathcal{P} \) satisfies a formula \( \varphi \) in a state \( s \) according to assignment \( \sigma \), or \( (\mathcal{P}, s, \sigma) \models \varphi \), as follows (clauses for propositional connectives are omitted as straightforward):

\[
(\mathcal{P}, s, \sigma) \models P(\bar{t}) \iff (\sigma(t_1), \ldots, \sigma(t_q)) \in D_s(\mathcal{P})
\]

\[
(\mathcal{P}, s, \sigma) \models t = t' \iff \sigma(t) = \sigma(t')
\]

\[
(\mathcal{P}, s, \sigma) \models \forall x \varphi \iff \text{for all } u \in \text{dom}(s) \cup \text{ag}(s), (\mathcal{P}, s, \sigma_u) \models \varphi
\]

\[
(\mathcal{P}, s, \sigma) \models AX \varphi \iff \text{for all } s' \text{-runs } r, (\mathcal{P}, r(1), \sigma) \models \varphi
\]

\[
(\mathcal{P}, s, \sigma) \models A \varphi U \varphi' \iff \text{for all } s' \text{-runs } r, (\mathcal{P}, r(k), \sigma) \models \varphi' \text{ for some } k \geq 0, \text{ and for all } j,
\]

\[
0 \leq j < k \text{ implies } (\mathcal{P}, r(j), \sigma) \models \varphi
\]

\[
(\mathcal{P}, s, \sigma) \models E \varphi U \varphi' \iff \text{for some } s' \text{-run } r, \text{ for some } k \geq 0,
\]

\[
(\mathcal{P}, r(k), \sigma) \models \varphi', \text{ and for all } j,
\]

\[
0 \leq j < k \text{ implies } (\mathcal{P}, r(j), \sigma) \models \varphi
\]

\[
(\mathcal{P}, s, \sigma) \models K_s \varphi \iff \text{for all } s' \in S, \ s \sim_s s' \Rightarrow (\mathcal{P}, s', \sigma) \models \varphi
\]

\[
(\mathcal{P}, s, \sigma) \models K_{ \alpha } \varphi \iff \text{for all } s' \in S, \ s \sim_\alpha s' \Rightarrow (\mathcal{P}, s', \sigma) \models \varphi
\]

A formula \( \varphi \) is true at \( s \), or \( (\mathcal{P}, s) \models \varphi \), if \( (\mathcal{P}, s, \sigma) \models \varphi \) for all assignments \( \sigma; \ \varphi \) is true in \( \mathcal{P}, \) or \( \mathcal{P} \models \varphi, \) if \( (\mathcal{P}, s_0) \models \varphi \) for all \( s_0 \in I \). We remark that Def. 9 adopts an active domain semantics, where quantifiers range over the set \( \text{dom}(s) \cup \text{ag}(s) \) of active individuals and agents. This is an extension to agents of the standard assumption in database theory, also used in data-aware systems [Belardinelli et al., 2014; Hariri et al., 2013]. Also, notice that the active domain may vary at each state. Furthermore, by definition of epistemic indistinguishability, if \( a \notin \text{ag}(s) \) then \( (\mathcal{P}, s, \sigma) \models K_a \varphi \), for all formulas \( \varphi \), as for no \( s' \in S, s' \sim_a s \). In other words, epistemic formulas are vacuously true for agents not in the active domain of the state considered. So, for an epistemic formula not to be satisfied, it is required that an agent in the active domain does not know the fact in question.

Finally, we state the model checking problem for OMAS with respect to the specification language FO-CTLK_\( \pi \).

**Definition 10 (Model Checking Problem)** Given an OMAS \( \mathcal{P} \) and an FO-CTLK_\( \pi \) formula \( \varphi \), determine whether for every initial state \( s_0 \in I, (\mathcal{P}, s_0, \sigma_0) \models \varphi \) for some assignment \( \sigma_0 \).

Def. 10 assumes that the transition function \( \tau \) is given as a computable function, and that we have finitary descriptions for the set \( I \) of initial states and the domain \( X \). These requirements are normally fulfilled in cases of interest (see Section 2.3). Moreover, the specification \( \varphi \) is typically an FO-CTLK_\( \pi \) sentence, with no free variables. Hence, the model checking problem reduces to determine whether \( \mathcal{P} \models \varphi \). Model checking general data-aware systems is known to be undecidable [Deutsch et al., 2007]. In [Belardinelli et al., 2012; 2014] this problem is proved decidable for bounded and uniform systems. However, all these contributions assume that the set of agents is fixed at design time. In [Belardinelli and Grossi, 2015] preliminary results on the verification of a particular class of OMAS are presented, but without considering the epistemic dimension.

### 2.3 Use Case: Knowledge in open MAS networks

We now illustrate the formalism introduced by means of an example on agent networks. In [Belardinelli and Grossi, 2015] it is shown how a non-probabilistic variant of the SIR network diffusion model (see [Jackson, 2008, Ch. 7]) can be formally verified against first-order, purely temporal specifications. In the SIR model a group of agents connected in a network structure goes through three different stages during an ‘epidemic’ involving the spread of diseases, ideas, information, or similar social phenomena. First, each agent is susceptible to be infected; she may actually get infected at a certain point depending on whether any of her neighbors in the network are also infected; then an agent will eventually recover. OMAS can be used to encode open and dynamic SIR models, also incorporating the epistemic aspects of diffusion. The specification language FO-CTLK_\( \pi \) allows us to express properties of SIR models concerning: (i) how knowledge influences diffusion through the network; and (ii) how knowledge itself spreads within the system.

Let a binary predicate \( N \) denote the network structure, so \( N(x, y) \) means that agents \( x \) and \( y \) are connected; while the unary predicates \( \text{Sus}, \text{Inf} \) and \( \text{Rec} \) denote the properties of being susceptible, infected, and recovered respectively. As examples of the first group of properties consider the following formulas:

\[
AG \forall x, y (K_x (\text{Inf}(y) \land N(x, y)) \rightarrow AF \text{--~} N(x, y)) \quad (2)
\]

\[
AG \forall x (K_x \text{Sus}(x) \rightarrow AF \forall y (N(x, y) \rightarrow \text{Rec}(y))) \quad (3)
\]

Formula (2) states that it is always the case that if an agent \( x \) knows that she is connected to an infected agent \( y \), then she will part at some point in the future. Formula (3) states that it is always the case that if an agent \( x \) knows she is susceptible, then eventually she will always be connected only to recovered agents.

We stress the fundamental difference between a quantified formula \( \forall x K_x \phi \), which express dynamically the joint knowledge of \( \phi \) for all active agents in a given state \( s \), and the standard, static epistemic formula \( E \phi \). Actually, for \( E \phi \) to be a formula, the set \( Ag \) of agents has to be finite and specified at design time. Moreover, a formula such as \( AG \forall x K_x \phi \) refers to the knowledge of a possibly different group of active agents at each time.

As examples of the second group of properties above con-
Consider the following formulas:

\[ AG \forall x (\text{Rec}(x) \rightarrow AF \exists y K_y \text{Rec}(x)) \]  
\[ AG \forall y (\text{Inf}(y) \rightarrow (AF \exists x (N(x, y) \rightarrow K_x \text{Inf}(y)))) \]

(4)  
(5)

Formula (4) states that it is always the case that if an agent is recovered, then this fact won’t be ignored, i.e., someone will know it. Formula (5) states that it is always the case that if some agent y is infected, then all agents that are connected to y will eventually know this fact. We stress once more that to express epistemic properties, such as (2)-(5) above, in open MAS we do need epistemic modalities indexed by terms and quantification, as the set \( Ag \) of agents is infinite in general.

In the next section we develop techniques to model check OMAS against such first-order temporal-epistemic specifications.

3 Bisimulation

In Section 2 we stated that model checking OMAS against FO-CTL\(_x\) specifications is undecidable in general. To single out semantical fragments with a decidable model checking problem, we first introduce a notion of bisimulation and show that bisimilar OMAS satisfy the same FO-CTL\(_x\) formulas. The results presented in this section build upon previous work in the literature [Belardinelli et al., 2012; 2014; Belardinelli and Grossi, 2015]. However, the present setting differs, as we consider open MAS, where agents can join and leave at run time, and our specification language contains term-indexed epistemic modalities.

In the rest of the paper we let \( \mathcal{P} = \langle Ag, U, I, \tau \rangle \) and \( \mathcal{P}' = \langle Ag', U', I', \tau' \rangle \) be OMAS and assume that \( s = \langle l_0, \ldots, l_n \rangle \in S \) and \( s' = \langle l'_0, \ldots, l'_n \rangle \in S' \). According to Def. 7 isomorphic states have the same relational structure, but to account also for values assigned to free variables we introduce the following notion.

**Definition 11 (Equivalent assignments)** Given states \( s \in S \) and \( s' \in S' \), and a formula \( \phi \), assignments \( \sigma : \text{Var} \rightarrow X \) and \( \sigma' : \text{Var} \rightarrow X' \) are equivalent for \( \phi \) (w.r.t. \( s \) and \( s' \)) iff for some bijection \( \gamma : \text{dom}(s) \cup \text{ag}(s) \cup \text{Con} \cup \sigma'(fr(\phi)) \rightarrow \text{dom}(s') \cup \text{ag}(s') \cup \text{Con} \cup \sigma'(fr(\phi)) \) (i) the restriction \( \gamma|_{\text{dom}(s) \cup \text{ag}(s) \cup \text{Con}} \) is a witness for \( s \equiv s' \); and (ii) \( \sigma'|_{fr(\phi)} = \gamma \circ \sigma(\phi) \).

Equivalent assignments preserve agent types, the (in)equalities in \( \phi \), as well as the active elements in \( s \) and \( s' \), modulo renaming.

Bisimilarities are known to preserve the satisfaction of modal formulas in a propositional setting [Blackburn et al., 2001, Ch. 2]. We now investigate under which conditions this is true of OMAS as well.

**Definition 12 (Simulation)** A relation \( R \subseteq S \times S' \) is a simulation iff \( R(s, s') \) implies (i) \( s \equiv s' \); (ii) for every \( t \in S \), if \( s \rightarrow t \) then for some \( t' \in S' \), \( s' \rightarrow t' \), \( s \equiv t \equiv \phi \circ t' \), and \( R(t, t') \); and (iii) for every \( t \in S \), if \( a \in \text{ag}(s) \), if \( s \rightarrow a \) then for some \( t' \in S' \), \( s' \rightarrow a \), \( s \equiv t' \equiv \phi \circ t' \), and \( R(t, t') \).

A state \( s' \) simulates \( s \) iff \( R(s, s') \) holds for some simulation \( R \). In particular, similar states are isomorphic by condition 12.1(i) above. Simulations can then be extended to bisimulations.

**Definition 13 (Bisimulation)** A relation \( B \subseteq S \times S' \) is a bisimulation iff both \( B \) and \( B^{-1} = \{ (s', s) \mid (s, s') \in B \} \) are simulations.

Two states \( s \) and \( s' \) are bisimilar, or \( s \equiv s' \), iff \( B(s, s') \) holds for some bisimulation \( B \). Notice that \( \equiv \) is the largest bisimulation and an equivalence relation on \( S \cup S' \). Finally, the OMAS \( \mathcal{P} \) and \( \mathcal{P}' \) are bisimilar, or \( \mathcal{P} \equiv \mathcal{P}' \), iff (i) for every \( s_0 \in I, s_0 \equiv s'_0 \) for some \( s'_0 \in I' \), and (ii) for every \( s'_0 \in I' \), \( s_0 \equiv s'_0 \) for some \( s_0 \in I \).

In [Belardinelli et al., 2014] it is shown that, differently from the propositional modal case, in data-aware systems bisimilarity does not preserve first-order temporal-epistemic formulas. Nonetheless, we prove that uniform OMAS admit FO-CTL\(_x\)-preserving bisimulations.

**Definition 14 (Uniformity)** An OMAS \( \mathcal{P} \) is uniform iff for every \( s, t, s', t' \in S \), \( t' \in T \), (i) if \( s \rightarrow t \) and \( s' \equiv t' \) then \( s' \rightarrow t' \); and (ii) for every \( a \in \text{ag}(s) \), if \( s \rightarrow_a t \) and \( s' \equiv t' \rightarrow_a t' \) then \( s' \rightarrow_a t' \).

Intuitively, uniformity expresses a fullness condition on OMAS: a uniform OMAS allows all “isomorphic” transitions. We discuss uniformity in more depth in Section 3.

We finally prove that FO-CTL\(_x\) formulas cannot distinguish between bisimilar and uniform OMAS, as long as specific cardinality constraints on the interpretation domains are satisfied.

**Theorem 1** Consider bisimilar and uniform OMAS \( \mathcal{P} \) and \( \mathcal{P}' \), bisimilar states \( s \in S \) and \( s' \in S' \), an FO-CTL\(_x\) formula \( \varphi \), and assignments \( \sigma \) and \( \sigma' \) equivalent for \( \varphi \) w.r.t. \( s \) and \( s' \).

1. for every \( s \)-run \( r \), for every \( k \geq 0 \), (i) \( |X'| \geq |\text{dom}(r(k)) \cup \text{ag}(r(k)) \cup \text{adom}(r(k + 1)) \cup \text{ag}(r(k + 1)) \cup \text{Con} \cup \sigma'(fr(\varphi))| + |\text{var}(\varphi) \setminus \text{fr}(\varphi)| 
\text{and (ii) } |\text{Adom}(r(k)) \cup \text{ag}(r(k + 1)) \cup \text{Con} \cup \sigma'(fr(\varphi)) + |\text{var}(\varphi) \setminus \text{fr}(\varphi)| \text{for every type } T; \)

2. for every \( s' \)-run \( r' \), for every \( k \geq 0 \), (i) \( |X| \geq |\text{dom}(r'(k)) \cup \text{ag}(r'(k)) \cup \text{dom}(r'(k + 1)) \cup \text{ag}(r'(k + 1)) \cup \text{Con} \cup \sigma'(fr(\varphi))| + |\text{var}(\varphi) \setminus \text{fr}(\varphi)| 
\text{and (ii) } |\text{Adom}(r'(k)) \cup \text{ag}(r'(k + 1)) \cup \text{Con} \cup \sigma'(fr(\varphi)) + |\text{var}(\varphi) \setminus \text{fr}(\varphi)| \text{ for every type } T; 
\text{then } (\mathcal{P}, s, \sigma) \models \varphi \Leftrightarrow (\mathcal{P}', s', \sigma') \models \varphi. 

As a consequence of Theorem 1, bisimilar states satisfy the same FO-CTL\(_x\) formulas for equivalent assignments, whenever cardinality constraints (1) and (2) are satisfied.

We now apply Theorem 1 to the model checking problem for OMAS. First of all, we introduce bounded OMAS.

**Definition 15 (Bounded OMAS)** An OMAS \( \mathcal{P} \) is \( b \)-bounded, for \( b \in \mathbb{N} \), iff for all \( s \in S \), \( |\text{adom}(s) \cup \text{ag}(s)| \leq b. \)

An OMAS \( \mathcal{P} \) is bounded iff it is \( b \)-bounded for some \( b \in \mathbb{N} \). We remark that bounded OMAS are still infinite-state systems in general. Hereafter let \( \text{sup}_{s \in S} \{|\text{adom}(s) \cup \text{ag}(s)|\} \) be equal to \( \infty \) whenever the OMAS \( \mathcal{P} \) is unbounded. Similarly for \( \text{sup}_{s \in S} \{|\text{adom}(s)\}| \).}

**Corollary 2** Consider bisimilar and uniform OMAS \( \mathcal{P} \) and \( \mathcal{P}' \), and an FO-CTL\(_x\) formula \( \varphi \). If
1. \[ |X'| \geq 2 \sup_{s \in S} \{|\text{dom}(s) \cup \text{ag}(s)|\} + |\text{con}(\phi)| + |\text{var}(\phi)| \quad \text{and} \quad |A_{g'}| \geq 2 \sup_{s \in S} \{|\text{ag}(s')|\} + |\text{con}(\phi)| + |\text{var}(\phi)| \quad \text{for every type } T; \]
2. \[ |X| \geq 2 \sup_{s \in S} \{|\text{dom}(s') \cup \text{ag}(s')|\} + |\text{con}(\phi)| + |\text{var}(\phi)| \quad \text{and} \quad |A_T| \geq 2 \sup_{s \in S} \{|\text{ag}(s')|\} + |\text{con}(\phi)| + |\text{var}(\phi)| \quad \text{for every type } T; \]

then \( P \models \phi \iff P' \models \phi. \)

Corollary 2 shows that an infinite-state OMAS \( P \) can in principle be verified by model checking a bisimilar system \( P' \), as long as \( X' \) is sufficiently large for \( P' \) to bisimulate \( P \).

In the next section we show that finite abstractions can indeed be defined for bounded OMAS, thus allowing for the verification of properties, such as (2)-(5) in Section 2.3, but first we briefly discuss uniform OMAS.

**Discussion: Uniformity** The notion of uniformity was put forward in Belardinelli et al., 2012 to prove the decidability of model checking data-aware systems. In Belardinelli et al., 2014 uniformity is related to genericity in databases [Abiteboul et al., 1995]. Intuitively, in uniform systems transitions depend on the logical form of data, rather than on the actual data content. It has been argued that the class of uniform systems covers most cases of interest [Hariri et al., 2013; Deutsch et al., 2007].

We now analyse uniformity in the context of OMAS through the following result.

**Lemma 3** Suppose that an OMAS \( P \) satisfies the following:

\[ \text{for } s_0 \in S, s'_0 \in S', s_0 \simeq s'_0 \text{ implies } s_0 \in I \iff s'_0 \in I \]  

\( (*) \)

Then, \( P \) is uniform.

The proof of Lemma 3 makes essential use of conditions \((*)\) and \((+)*\) on protocols and transition functions respectively. Thus, requirements (i) and (ii) in Def. 14 can be substituted by closure \((*)\) of initial states under isomorphism. The latter condition is quite natural, as we are mainly interested in the relational structure of data, not the actual data content. In the final section we assume that all OMAS satisfy \((*)\) and are therefore uniform.

### 3.1 Finite Abstraction

We now show that an infinite OMAS can in principle be verified by checking a finite abstraction. The main result is Theorem 5, which ensures that boundedness and closure under uniform initial states \((*)\) are sufficient to obtain finite bisimilar abstractions, thus preserving FO-CTL\(_x\) formulas.

We first define the notion of abstract agents.

**Definition 16 (Abstract agents)** Let \( a = \langle \mathcal{D}, \text{Act}, Pr \rangle \in Ag_T \) be an agent of type \( T \) defined on a domain \( X = U \cup Ag \). Given a set \( X' = U' \cup Ag' \) of elements, the abstract agent \( a' \) is the tuple \( \langle \mathcal{D}, \text{Act}, Pr' \rangle \) on \( X' \) s.t. \( Pr' \) is the smallest function defined as

- \( \alpha(i) \in Pr(I), l' \in D'(X') \) and \( l' \rightharpoonup l \), then \( \alpha(i)(i) \in Pr'(l') \).

Given a set \( Ag_T \) of agents, let \( Ag' \) be the set of the corresponding abstract agents. Notice that \( Ag \) and \( Ag' \) are used to denote both the set of agent names and of agents; the context will disambiguate. The abstract agent \( a' \) in Def. 16 is indeed an agent of type \( T \), as defined in Def. 4, since \( a \) and \( a' \) share the same database schema and actions. Moreover, protocol \( Pr' \) is well-defined whenever \( Pr \) is, and it satisfies condition \((*)\) on protocols by definition. We now present abstractions.

**Definition 17 (Abstractions)** Let \( P = \langle Ag, U, I, \tau \rangle \) be an OMAS, and \( Ag' \) the set of abstract agents defined on \( X' \) as in Def. 16. The OMAS \( P' = \langle Ag', U', I', \tau' \rangle \) is an abstraction of \( P \) if (i) \( I' = \{s'_0 \in G' \mid \exists s_0 \text{ s.t. } s_0 \simeq s'_0 \text{ for some } s_0 \in I\} \), and (ii) \( \tau' \) is the smallest function defined as follows

- if \( s \xrightarrow{\alpha(i)} t \) in \( P \), \( s', t' \in G' \), and \( s + t \simeq s' + t' \) for some witness \( i \), then \( s' \xrightarrow{\alpha(i)} t' \).

The abstraction \( P' \) in Def. 17 is an OMAS as it complies with Def. 6. Moreover, condition \((+)*\) on transition functions is satisfied. Notice that, by varying \( X' \) we can obtain abstractions of different cardinalities, in particular finite abstractions.

Next, we explore the relationship between an OMAS and its abstractions. By the next result every abstraction is uniform, independently from the concrete OMAS.

**Lemma 4** Every abstraction \( P' \) of an OMAS \( P \) is uniform. Moreover, if \( P \) is uniform and \( X' = X \), then \( P' = P \).

By the next result there exists a bisimilar abstraction for every bounded OMAS, provided that the former is built over a sufficiently large domain. Hereafter we suppose that, for a bound \( b \in \mathbb{N}, N_b \) is the maximum numbers of parameters contained in any parametric joint actions, i.e., \( N_b = b \cdot \max\{|\alpha(i)| \in \text{Act}_T, \text{type } T\} \).

**Theorem 5** Consider a bounded OMAS \( P \) over an infinite domain \( X \), an FO-CTL\(_x\) formula \( \varphi \), and a domain \( X' \supseteq \text{con}(\varphi) \). If (i) \( |X'| \geq 2b + |\text{con}(\varphi)| + \max\{|\text{var}(\varphi)|, N_b\} \), and (ii) \( \text{for every type } T, |A_{g'}| \geq 2b + |\text{con}(\varphi)| + \max\{|\text{var}(\varphi)|, N_b\} \), then there exists a bisimilar abstraction \( P' \) of \( P \) over \( X' \). In particular, \( P \models \varphi \iff P' \models \varphi \).

Notice that each \( Ag' \) and \( X' \) in Theorem 5 might as well be finite. So, by using a sufficient number of abstract agents and values, we can in principle reduce the model checking problem for infinite-state OMAS to the verification of a finite abstraction. Specifically, we obtain the following corollary to Theorem 5.

**Corollary 6** Given a bounded OMAS \( P \) over an infinite domain \( X \), and an FO-CTL\(_x\) formula \( \varphi \), there exists an abstract OMAS \( P' \) over a finite domain \( X' \) s.t. \( \varphi \) holds in \( P \) iff it holds in \( P' \).

As a consequence of Corollary 6, we can in principle verify an infinite-state, bounded OMAS, by model checking its finite, bisimilar abstraction.

### 4 Conclusions

In this paper we addressed the formal verification of MAS where an infinite number of agents may in principle be present and may be entering and leaving the system at run time. A notable feature of our proposal concerns the specification language FO-CTL\(_x\), which includes epistemic operators indexed by individual terms. As we discussed, the latter are key to express relevant properties of OMAS. We analysed
the model checking problem in this setting, and showed that it can be addressed through finite bisimilar abstractions, under some natural conditions.

An open problem not tackled in the present contribution and left for future work is the development of methodologies for generating finite abstractions, so that effective model checking procedures can be provided. This is a major challenge for the verification of open MAS.

Acknowledgements The research described in this paper was partly funded by the EPSRC Research Project “Trusted Autonomous Systems” (grant No. EP/I00520X/1).

References


