An Abstraction Technique for the Verification of Artifact-Centric Systems

Francesco Belardinelli
Department of Computing
Imperial College London
f.belardinelli@imperial.ac.uk

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

Fabio Patrizi
Dipartimento di Ingegneria Informatica,
Automatica e Gestionale "A. Ruberti"
Sapienza Università di Roma
patrizi@dis.uniroma1.it

Abstract

We explore the paradigm of artifact-centric systems from a knowledge-based perspective. We provide a semantics based on interpreted-systems to interpret a first-order temporal-epistemic language with identity in a multi-agent setting. We consider the model checking problem for this language and provide abstraction results. We isolate a natural subclass of artifact-systems for which the model checking problem is decidable. We give an upper bound on the complexity of the model checking problem.

1 Introduction

Much of the work in the area of *reasoning about knowledge* involves the development of formal techniques for the representation and automatic deduction of epistemic properties of a system. The approaches based on modal logic are often rooted in the seminal interpreted systems semantics (Parikh and Ramanujam 1985) for the interpretation of the standard multi-agent epistemic logic $\mathrm{S5}_n$. This approach was thoroughly explored in the 1990s leading to the significant body of work summarised in (Fagin et al. 1995). While the emphasis is on epistemic properties resulting from various forms of interaction, these are grounded on the AI notion of agents interacting in a multi-agent system. Indeed, some of the more recent work in this area (Gammie and van der Meyden 2004; Kacprzak et al. 2008; Lomuscio, Qu, and Raimondi 2009) has focused on the development of automatic techniques, including model checking (Clarke, Grumberg, and Peled 1999) for the verification of the epistemic properties of multi-agent systems. This has led to applications in a number of areas, traditionally outside AI and knowledge representation, including security protocols (Boureanu, Cohen, and Lomuscio 2009) and cache-coherence protocols (Baukus and van der Meyden 2004).

The ambition of the present paper is to begin to offer a similar change of perspective in a growing topic in web-services: *artifact systems* (Cohn and Hull 2009). Differently from the standard process-based paradigm popular in web-services, in artifact systems data are given the same prominence as processes. Implementations of artifact systems involve the composition of several databases on which services operate. This emphasis on data makes the automata-based formalisms commonly used to model services generally insufficient: one really requires to account for the evolution of the first-order schemas representing the underlying databases. While this makes the resulting verification problem for artifact systems undecidable, partial solutions have been put forward (Deutsch et al. 2009; Belardinelli, Lomuscio, and Patrizi 2011b; 2011a). These techniques enable the verification of basic temporal properties of the artifact system when analysed under a number of conditions.

While these results are valuable, they concern the verification of properties of the artifact system *per se* and do not address the need of modelling and verifying the actions and information properties of the *services acting on or through the artifact system.* But if artifact systems are to be deployed in a range of information services as their proponents recommend, it is paramount we can model what actions specific participants are allowed and not allowed to make, what knowledge they can and cannot derive in a system run, what system state they can achieve in coordination with their peers, etc. In other words, we need to verify the epistemic properties of a multi-agent system in which artifacts are present, and not just the evolution of the artifacts themselves.

In this paper we set about to explore the verification problem for artifact-centric multi-agent systems, i.e., systems of agents interacting through artifact systems, by means of a knowledge-based perspective. We operate in a first-order setting with knowledge and branching time (FO-CTLK). We first remark that the general problem is undecidable and then proceed to give abstraction results that enable us, in a large class of cases of practical interest, to be able to analyse the model checking of finite approximations rather than the plain infinite model with unbounded database schemas. We stress the fact that all results are obtained for the full first-order temporal epistemic logic FO-CTLK with identity. So, we assume no restriction on the specification language.

The rest of the paper is as follows. In Section 2 we identify the problem, give a general semantics for artifact-centric multi-agent systems (AC-MAS), as well as comment on the undecidability of the model checking problem in the general case. In Section 3 we explore abstraction results that enable us to translate the problem to the model checking of finite models. In Section 4 we take a step towards implementations and instantiate this semantics by means of artifact system descriptions that are close to what used in practice. We show that the setting complies with the conditions explored in the previous section thereby enabling us to show results for artifact-centric implementations. We conclude by presenting complexity results for the model checking problem and a discussion on related work.

2 Verification of AC-MAS

In this section we lay out the semantics of artifact-centric multi-agent systems, we then define an interpretation for a first-order temporal epistemic specification language, and introduce the model checking problem for these systems. We use basic definitions taken from (Belardinelli, Lomuscio, and Patrizi 2011b) to fix the notation used in the rest of the paper. We refer to the cited work for more details.

Databases and First-Order Logic

We now introduce the syntax and semantics of our first-order language. We also fix the terminology on databases that will be used in the rest of the paper (Abiteboul, Hull, and Vianu 1995).

Definition 2.1 (Database schema) *A* (relational) database schema *is a set* $\mathcal{D} = \{P_1/q_1, \ldots, P_n/q_n\}$ *of* relation symbols *P_i, each associated with its arity $q_i \in \mathbb{N}$.*

The *primed version* of a database schema $\mathcal{D}$ is the schema $\mathcal{D}' = \{P_1'/q_1, \ldots, P_n'/q_n\}$, where each predicate symbol P_i is (syntactically) replaced by its *primed version* P_i'. The motivation for $\mathcal{D}'$ will become apparent later on, as $\mathcal{D}$ and $\mathcal{D}'$ are used to account for the evolution of the artifact system.

Definition 2.2 (Database interpretation) *Given a database schema $\mathcal{D}$, a $\mathcal{D}$*-interpretation (or $\mathcal{D}$-instance) over an interpretation domain U *is a mapping D associating each relation symbol P_i with a* finite *q_i-ary relation over U, i.e., $D(P_i) \subseteq U^{q_i}$.*

The set of all $\mathcal{D}$-interpretations over a domain U is denoted by $\mathcal{D}(U)$. The *active domain* of D, or $ad(D)$, is the set of all individuals in U occurring in some tuple of some predicate interpretation $D(P_i)$. Observe that, since $\mathcal{D}$ contains a finite number of relation symbols and each $D(P_i)$ is finite, so is $ad(D)$.

Definition 2.3 (FO-formulas over $\mathcal{D}$) *Given a set Var of* individual variables *and a set $Con \subseteq U$ of* individual constants, *the formulas φ in the first-order language $\mathcal{L}_{\mathcal{D}}$ over $\mathcal{D}$ are defined in BNF as follows:*

$$\varphi \quad ::= \quad t = t' \mid P_i(\vec{t}) \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \forall x \varphi$$

where $\vec{t}$ is a q_i-tuple of terms, *and t, t' are* terms, *i.e., elements in $Var \cup Con$.*

Notice that formulas in $\mathcal{L}_{\mathcal{D}}$ contain no function symbols apart from constants. We use the standard abbreviations $\exists$, $\top$, $\bot$, $\wedge$, $\vee$, and $\neq$. Free and bound variables are defined as standard. For a formula φ we denote the set of its variables as $var(\varphi)$, the set of its free variables as $free(\varphi)$, and the set of constants in φ as $con(\varphi)$. We write $\varphi(\vec{x})$ to list explicitly in arbitrary order all the free variables $x_1, \ldots, x_\ell$ of φ. By slight abuse of notation, we treat $\vec{x}$ as a set, thus we write $\vec{x} = free(\varphi)$. A *sentence* is a formula with no free variables.

An *assignment* is a function $\sigma : Var \mapsto U$. For an assignment σ, we denote by $\sigma\binom{x}{u}$ the assignment s.t. (i) $\sigma\binom{x}{u}(x) = u$; and (ii) $\sigma\binom{x}{u}(x') = \sigma(x')$ for every $x' \in Var$ different from x. We extend assignments to constants by assuming that $\sigma(c) = c$ for all $c \in Con$.

Definition 2.4 (Semantics of FO-formulas) *Given a $\mathcal{D}$-interpretation D, an assignment σ, and an FO-formula $\varphi \in \mathcal{L}_{\mathcal{D}}$, we inductively define whether D satisfies φ under σ, written $(D, \sigma) \models \varphi$, as follows:*

$$\begin{array}{lll}
(D,\sigma) \models P_i(\vec{t}) & \textit{iff} & \langle \sigma(t_1), \ldots, \sigma(t_{q_i}) \rangle \in D(P_i) \\
(D,\sigma) \models t = t' & \textit{iff} & \sigma(t) = \sigma(t') \\
(D,\sigma) \models \neg\varphi & \textit{iff} & (D,\sigma) \not\models \varphi \\
(D,\sigma) \models \varphi \rightarrow \psi & \textit{iff} & (D,\sigma) \not\models \varphi \textit{ or } (D,\sigma) \models \psi \\
(D,\sigma) \models \forall x \varphi & \textit{iff} & \textit{for all } u \in ad(D),\ (D, \sigma\binom{x}{u}) \models \varphi
\end{array}$$

A formula φ is true *in D, written $D \models \varphi$, iff $(D,\sigma) \models \varphi$ for all assignments σ.*

Observe that we are adopting an *active-domain* semantics, that is, quantified variables range over the active domain of D. Also notice that, by definition of σ, constants are interpreted as themselves. Thus, two constants are equal iff they are syntactically the same.

As an example of an FO-formula consider

$$\phi_k \quad ::= \quad \forall x_1, \ldots, x_{k+1} \bigvee_{i \neq j} (x_i = x_j)$$

which intuitively says that there are at most k distinct elements. Such formula will be used in Sec. 5.

Finally, for technical purposes, we define the operator $\oplus$.

Definition 2.5 ($\oplus$ Operator) *Given two $\mathcal{D}$-interpretations D and D', $D \oplus D'$ is the $(\mathcal{D} \cup \mathcal{D}')$-interpretation s.t. $D \oplus D'(P_i) = D(P_i)$ and $D \oplus D'(P_i') = D'(P_i)$.*

The $\oplus$ operator will be used later in connection with the transitions of artifact systems.

Artifact-Centric Multi-Agent Systems

We follow the construction in (Belardinelli, Lomuscio, and Patrizi 2011a) to define an extension of interpreted systems to first order, which allows us to reason in terms of the local database schemas of agents. We refer to (Fagin et al. 1995) for background on the interpreted systems semantics.

Definition 2.6 (Agent) *An* agent *is a tuple $i = \langle \mathcal{D}_i, L_i, Act_i, Pr_i \rangle$ where:*

- *$\mathcal{D}_i$ is the* local database schema*;*
- *$L_i \subseteq \mathcal{D}_i(U)$ is the set of* local states*;*
- *Act_i is the set of* local actions*;*
- *$Pr_i : L_i \mapsto 2^{Act_i}$ is the* local protocol function.

Given a set $Ag = \{1, \ldots, n\}$ of agents, we define the *global* database schema of Ag as $\mathcal{D} \doteq \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_n$, i.e., the set of all relation symbols appearing in some local database schema. Agents can be thought of as modules that can be composed together to obtain artifact-centric multi-agent systems.

Definition 2.7 (AC-MAS) *Given a set Ag of agents, an* artifact-centric multi-agent system *is a tuple $\mathcal{P} = \langle \mathcal{S}, U, D_0, \tau \rangle$ where:*

- *$\mathcal{S} \subseteq L_1 \times \cdots \times L_n$ is the set of* reachable global states*;*
- *U is the interpretation domain;*
- *$D_0 \in \mathcal{S}$ is the* initial global state*;*
- *$\tau : \mathcal{S} \times Act \mapsto 2^{\mathcal{S}}$ is the* global transition function*, where $Act = Act_1 \times \cdots \times Act_n$ is the set of* global actions*, and $\tau(\langle l_1, \ldots, l_n \rangle, \langle \alpha_1, \ldots, \alpha_n \rangle)$ is defined iff $\alpha_i \in Pr_i(l_i)$ for every $i \in Ag$.*

Given an AC-MAS $\mathcal{P}$, we can interpret a global state $\langle l_1, \ldots, l_n \rangle$ as the $\mathcal{D}$-instance D s.t. $D(P_i) = \bigcup_{j \in Ag} l_j(P_i)$, for all $P_i \in \mathcal{D}$. Thus, the set $\mathcal{S}$ of global states can be seen as a subset of $\mathcal{D}(U)$. Notice that for each $\langle l_1, \ldots, l_n \rangle$ there exists a unique $\mathcal{D}$-instance D as above, however the converse is not true in general. The way D has to be interpreted will be clear from the context.

We define the *global transition relation* $\rightarrow$ on $\mathcal{S} \times \mathcal{S}$ s.t. $D \rightarrow D'$ if there exists $\vec{\alpha} \in Act$ and $D' \in \tau(D, \vec{\alpha})$. If $D \rightarrow D'$, we say that D' is a *successor* of D. A *run* r from $D \in \mathcal{S}$ is an infinite sequence $D^0 \rightarrow D^1 \rightarrow \cdots$, for $D^0 = D$. For $n \in \mathbb{N}$, $r(n) \doteq D^n$. A state D' is *reachable from* D if there exists a run r from $r(0) = D$ s.t. $r(i) = D'$ for some $i \geq 0$. In what follows we assume that the relation $\rightarrow$ is serial, and that $\mathcal{S}$ is the set of states reachable from D_0.

We say that D, D' are *epistemically indistinguishable* for agent i, written $D \sim_i D'$, if $l_i(D) = l_i(D')$, where $l_i(D) \doteq l_i$ for $D = \langle l_1, \ldots, l_n \rangle$. This is consistent with the standard definition of knowledge as identity of local states (Fagin et al. 1995).

Model Checking

We now define the first-order temporal epistemic specification language to be interpreted on AC-MAS. Differently from (Belardinelli, Lomuscio, and Patrizi 2011b), we assume no restriction on the interaction between quantifiers and modalities.

Definition 2.8 (FO-CTLK) *The first-order CTLK formulas φ over a database schema $\mathcal{D}$ are inductively defined by the following BNF:*

$$\varphi ::= \phi \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \forall x \varphi \mid AX\varphi \mid A\varphi\mathcal{U}\varphi \mid E\varphi\mathcal{U}\varphi \mid K_i\varphi$$

where $\phi \in \mathcal{L}_{\mathcal{D}}$.

The notions of free and bound variables for FO-CTLK extend straightforwardly from $\mathcal{L}_{\mathcal{D}}$, as well as functions var, $free$, and con. We use the abbreviations $EX\varphi$, $AF\varphi$, $AG\varphi$, $EF\varphi$, and $EG\varphi$ as standard.

Definition 2.9 (Semantics of FO-CTLK formulas) *Consider an AC-MAS $\mathcal{P}$, an FO-CTLK formula φ, a state $D \in \mathcal{P}$, and an assignment σ. We inductively define whether $\mathcal{P}$* satisfies *φ in D under σ, written $(\mathcal{P}, D, \sigma) \models \varphi$, as follows:*

$$\begin{array}{ll}
(\mathcal{P}, D, \sigma) \models \varphi & \textit{iff } (D, \sigma) \models \varphi \textit{ and } \varphi \textit{ is an FO-formula} \\
(\mathcal{P}, D, \sigma) \models \neg\varphi & \textit{iff } (\mathcal{P}, D, \sigma) \not\models \varphi \\
(\mathcal{P}, D, \sigma) \models \varphi \rightarrow \varphi' & \textit{iff } (\mathcal{P}, D, \sigma) \not\models \varphi \textit{ or } (\mathcal{P}, D, \sigma) \models \varphi' \\
(\mathcal{P}, D, \sigma) \models \forall x \varphi & \textit{iff for all } u \in ad(D),\ (\mathcal{P}, D, \sigma\binom{x}{u}) \models \varphi \\
(\mathcal{P}, D, \sigma) \models AX\varphi & \textit{iff for all } r, \textit{ if } r(0) = D \textit{ then } (\mathcal{P}, r(1), \sigma) \models \varphi \\
(\mathcal{P}, D, \sigma) \models A\varphi\mathcal{U}\varphi' & \textit{iff for all } r, \textit{ if } r(0) = D \textit{ then there is } k \geq 0 \\
 & \quad \textit{s.t. } (\mathcal{P}, r(k), \sigma) \models \varphi', \textit{ and for all } j, \\
 & \quad 0 \leq j < k \textit{ implies } (\mathcal{P}, r(j), \sigma) \models \varphi \\
(\mathcal{P}, D, \sigma) \models E\varphi\mathcal{U}\varphi' & \textit{iff for some } r,\ r(0) = D \textit{ and there is } k \geq 0 \\
 & \quad \textit{s.t. } (\mathcal{P}, r(k), \sigma) \models \varphi', \textit{ and for all } j, \\
 & \quad 0 \leq j < k \textit{ implies } (\mathcal{P}, r(j), \sigma) \models \varphi \\
(\mathcal{P}, D, \sigma) \models K_i\varphi & \textit{iff } D \sim_i D' \textit{ implies } (\mathcal{P}, D', \sigma) \models \varphi
\end{array}$$

A formula φ is *true* at D, written $(\mathcal{P}, D) \models \varphi$, if $(\mathcal{P}, D, \sigma) \models \varphi$ for all σ; φ is *true* in $\mathcal{P}$, written $\mathcal{P} \models \varphi$, if $(\mathcal{P}, D_0) \models \varphi$.

In the rest of the paper we explore model checking of AC-MAS against first-order temporal epistemic specifications. Formally, the problem can be stated as follows:

> Given an AC-MAS $\mathcal{P}$ and an FO-CTLK formula φ, find an assignment σ such that $(\mathcal{P}, D_0, \sigma) \models \varphi$.

It was remarked in (Belardinelli, Lomuscio, and Patrizi 2011b) that, even in the context of a purely temporal language, the problem is decidable whenever the domain U in $\mathcal{P}$ is finite, and undecidable in general. Since we are here operating on a strictly stronger language, the following immediately follows.

Theorem 2.10 *The model checking problem for AC-MAS is undecidable.*

The result above is obviously negative. However, in the following we identify a large class of AC-MAS, covering interesting cases, for which model checking is decidable.

3 Abstraction of AC-MAS

In this section we present a technique for defining finite abstractions of AC-MAS, so that the satisfaction of FO-CTLK formulas is preserved. Unless differently stated, in the rest of the paper we assume fixed a finite set C of constants, and a countable interpretation domain U such that $C \subseteq U$. Further, whenever we consider an FO-CTLK formula φ, we assume that $con(\phi) \subseteq C$. This can be always done w.l.o.g. by extending C with the finitely many constants in φ. Finally, the $\mathcal{D}$-instances D_1 and D_2 are defined on the interpretation domains $U_1, U_2 \subseteq U$.

Bisimulation

To show that abstractions of AC-MAS preserve FO-CTLK formulas, we introduce a notion of *bisimulation*, which refines the one presented in (Belardinelli, Lomuscio, and Patrizi 2011b). First we define when two $\mathcal{D}$-instances are *isomorphic*.

Definition 3.1 (Isomorphism) *Two $\mathcal{D}$-instances D_1 and D_2 are* isomorphic*, written $D_1 \simeq D_2$, iff there exists a bijection $\iota : ad(D_1) \cup C \mapsto ad(D_2) \cup C$ s.t.*

(i) ι is the identity on C;

(ii) for every $P_j \in \mathcal{D}$, for every $\vec{u} \in ad(D_1)^{q_j}$, $\vec{u} \in D_1(P_j)$ iff $\iota(\vec{u}) \in D_2(P_j)$.

Isomorphisms of $\mathcal{D}$-instances preserve the constants in C and the interpretation of the relation symbols in $\mathcal{D}$. Any function ι as above is a *witness* for $D_1 \simeq D_2$. The relation $\simeq$ is obviously an equivalence relation. Given a function $f : U_1 \mapsto U_2$ that is total on $ad(D_1)$, $f(D_1)$ denotes the $\mathcal{D}$-instance over U_2 obtained from D_1 by renaming each of its elements $u \in ad(D_1)$ as $f(u)$. Observe that if f is also injective (thus invertible) on $ad(D_1)$, and is the identity on C, then $f(D_1) \simeq D_1$.

Being isomorphic is not a sufficient condition for two $\mathcal{D}$-instances to satisfy the same first-order formulas, as we have to take care also of free variable. Hence, we introduce the following notion.

Definition 3.2 (Equivalent assignments) *Let $V \subseteq Var$ be a set of variables, and assume $D_1 \simeq D_2$. Two assignments $\sigma_1 : Var \mapsto U_1$ and $\sigma_2 : Var \mapsto U_2$ are* equivalent for V (w.r.t. D_1 and D_2) *iff there exists a bijection $\gamma : ad(D_1) \cup C \cup \sigma_1(V) \mapsto ad(D_2) \cup C \cup \sigma_2(V)$ s.t.*

(i) $\gamma|_{ad(D_1)\cup C}$ is a witness for $D_1 \simeq D_2$;

(ii) $\sigma_2|_V = \gamma \circ \sigma_1|_V$.

Intuitively, this definition captures the fact that σ_1 and σ_2 preserve all (in)equalities of values assigned to variables in V, and the assignments are consistent with some witness ι for $D_1 \simeq D_2$. We say that two assignments are equivalent for a FO-CTLK formula φ if they are equivalent for $free(\varphi)$. We can now prove the following result on satisfaction preservation.

Proposition 3.3 *Given two isomorphic $\mathcal{D}$-instances D_1 and D_2, an FO-formula φ, and two assignments σ_1 and σ_2 equivalent for φ, we have that*

$$(D_1, \sigma_1) \models \varphi \quad \textit{iff} \quad (D_2, \sigma_2) \models \varphi$$

Proof (sketch). By induction on the structure of φ. □

Essentially, Prop. 3.3 says that isomorphic instances cannot be distinguished by FO-formulas. We aim to generalise this intuition, and define conditions that guarantee two AC-MAS to be indistinguishable by FO-CTLK formulas. To do so, we need the following central notions.

Definition 3.4 (Similarity) *Given two AC-MAS $\mathcal{P}_1 = \langle \mathcal{S}_1, U_1, D_{10}, \tau_1 \rangle$ and $\mathcal{P}_2 = \langle \mathcal{S}_2, U_2, D_{20}, \tau_2 \rangle$, $\mathcal{P}_2$* simulates *$\mathcal{P}_1$, written $\mathcal{P}_1 \preceq \mathcal{P}_2$, iff there exists a relation $R \subseteq \mathcal{S}_1 \times \mathcal{S}_2$, called* simulation relation*, s.t. $\langle D_{10}, D_{20} \rangle \in R$, and if $\langle D_1, D_2 \rangle \in R$ then:*

1. *$D_1 \simeq D_2$;*
2. *for every D_1', if $D_1 \rightarrow D_1'$ then there is D_2' s.t. $D_2 \rightarrow D_2'$, $D_1 \oplus D_1' \simeq D_2 \oplus D_2'$, and $\langle D_1', D_2' \rangle \in R$;*
3. *for every D_1', if $D_1 \sim_i D_1'$ then there is D_2' s.t. $D_2 \sim_i D_2'$, $D_1 \oplus D_1' \simeq D_2 \oplus D_2'$, and $\langle D_1', D_2' \rangle \in R$.*

Observe that R is well-defined as $D_1 \oplus D_1' \simeq D_2 \oplus D_2'$ implies $D_1' \simeq D_2'$. When $\langle D_1, D_2 \rangle \in R$ for some simulation relation R, we say that D_2 *simulates* D_1, written $D_1 \preceq D_2$. It can be proved that $\preceq$ is the largest simulation relation, w.r.t. set inclusion, and it is transitive.

Based on the notion of simulation, we can now define when two AC-MAS are *bisimilar*.

Definition 3.5 (Bisimilarity) *Two AC-MAS $\mathcal{P}_1$ and $\mathcal{P}_2$ are* bisimilar, *written $\mathcal{P}_1 \approx \mathcal{P}_2$, iff there exists a relation $B \subseteq \mathcal{S}_1 \times \mathcal{S}_2$, called* bisimulation relation*, s.t. both B and $B^{-1} = \{\langle D_2, D_1 \rangle \mid \langle D_1, D_2 \rangle \in B\}$ are simulation relations between $\mathcal{P}_1$ and $\mathcal{P}_2$.*

When $\langle D_1, D_2 \rangle \in B$ for some bisimulation relation B, we say that D_1 and D_2 are *bisimilar*, written $D_1 \approx D_2$. Obviously, if $D_1 \approx D_2$ then $D_1 \preceq D_2$ and $D_2 \preceq D_1$. However, the viceversa is not true. It can be easily proved that $\approx$ is an equivalence relation.

Finally, we introduce the class of AC-MAS of our interest, for which we prove invariance w.r.t. FO-CTLK formulas, and the main abstraction results.

Definition 3.6 (Uniformity) *An AC-MAS $\mathcal{P} = \langle \mathcal{S}, U, D_0, \tau \rangle$ is* uniform *iff for every $D, D', D'' \in \mathcal{S}$ and $D''' \in \mathcal{D}(U)$:*

1. *$D \rightarrow D'$ and $D \oplus D' \simeq D'' \oplus D'''$ imply $D'' \rightarrow D'''$;*
2. *$D \sim_i D'$ and $D \oplus D' \simeq D'' \oplus D'''$ imply $D'' \sim_i D'''$.*

Intuitively, uniformity requires that the transition and epistemic relations do not depend on the actual data content of each $\mathcal{D}$-instance (apart from the constants in C). Put differently, the requirements above capture the fact that the system is unable to distinguish among states containing the same constants and having the same data structure.

In particular, we can prove that under specific assumptions req. 2 holds for any AC-MAS.

Proposition 3.7 *If an AC-MAS $\mathcal{P}$ satisfies req. 1 in Def. 3.6 and $ad(D_0) \subseteq C$, then req. 2 is also satisfied.*

Proof (sketch). If $D \oplus D' \simeq D'' \oplus D'''$, then there is a witness $\iota : ad(D) \cup ad(D') \cup C \mapsto ad(D'') \cup ad(D''') \cup C$ that is the identity on C, hence also on $ad(D_0)$. Since $D \sim_i D'$, then $l_i(D) = l_i(D')$, and also $l_i(D'') = \iota(l_i(D)) = \iota(l_i(D')) = l_i(D''')$. Notice that this does not guarantee that $D'' \sim_i D'''$, as we need to prove that $D''' \in \mathcal{S}$. This can be done by showing that D''' is reachable from D_0. □

Prop. 3.7 identifies a sufficient condition for uniformity. However, in general AC-MAS are not uniform. We will return to this point in Sec. 4.

The rest of this section is dedicated to proving that bisimilarity together with uniformity are sufficient to preserve satisfaction of FO-CTLK formulas. We start by exploring the relationship between uniformity and bisimilarity.

Proposition 3.8 *If an AC-MAS $\mathcal{P}$ is uniform, then for every $D_1, D_2 \in \mathcal{S}$, $D_1 \simeq D_2$ implies $D_1 \approx D_2$.*

Proof (sketch). The proof consists in showing that $B = \{\langle D_1, D_2 \rangle \in \mathcal{S} \times \mathcal{S} \mid D_1 \simeq D_2\}$ is a bisimulation. □

The next results are aimed to show that if two bisimilar AC-MAS contain enough individuals in their interpretation domains, they satisfy the same FO-CTLK formulas.

Proposition 3.9 *Consider two bisimilar uniform AC-MAS $\mathcal{P}_1$ and $\mathcal{P}_2$, two $\mathcal{D}$-instances $D_1 \in \mathcal{P}_1$, $D_2 \in \mathcal{P}_2$ s.t. $D_1 \approx D_2$, and an FO-CTLK formula φ. For every assignments σ_1 and σ_2 equivalent for φ w.r.t. D_1 and D_2 we have that:*

1. *for every D'_1 s.t. $D_1 \rightarrow D'_1$, if $|U_2| \geq |ad(D_1) \cup ad(D'_1) \cup C \cup \sigma_1(free(\varphi))|$, then there exists D'_2 s.t. $D_2 \rightarrow D'_2$, $D'_1 \approx D'_2$, and σ_1 and σ_2 are equivalent for φ w.r.t. D'_1 and D'_2.*
2. *for every D'_1 s.t. $D_1 \sim_i D'_1$, if $|U_2| \geq |ad(D_1) \cup ad(D'_1) \cup C \cup \sigma_2(free(\varphi))|$, then there exists D'_2 s.t. $D_2 \sim_i D'_2$, $D'_1 \approx D'_2$, and σ_1 and σ_2 are equivalent for φ w.r.t. D'_1 and D'_2.*

Proof. To prove 1, let γ be a bijection witnessing that σ_1 and σ_2 are equivalent for φ w.r.t. D_1 and D_2, as in Def. 3.2. Consider D'_1 s.t. $D_1 \rightarrow D'_1$. Since $D_1 \approx D_2$, there exists a $D''_2 \in \mathcal{S}_2$ s.t. $D_2 \rightarrow D''_2$, $D_1 \oplus D'_1 \simeq D_2 \oplus D''_2$, and $D'_1 \approx D''_2$. Define $Dom(j) \doteq ad(D_1) \cup ad(D'_1) \cup C$, and partition it into:

- $Dom(\gamma) \doteq ad(D_1) \cup C \cup (ad(D'_1) \cap \sigma_1(free(\varphi))$;
- $Dom(\iota') \doteq ad(D'_1) \setminus Dom(\gamma)$;

and define the function $j : Dom(j) \mapsto U_2$ as follows:

$$j(u) = \begin{cases} \gamma(u), \text{ if } u \in Dom(\gamma) \\ \iota'(u), \text{ if } u \in Dom(\iota') \end{cases}$$

where $\iota' : Dom(i') \mapsto U_2 \setminus Im(\gamma)$ is any invertible (total) function It can be proved that $|U_2| \geq |ad(D_1) \cup ad(D'_1) \cup C \cup \sigma_1(free(\varphi))|$ implies $|U_2 \setminus Im(\gamma)| \geq |Dom(\iota')|$, thus such a ι' exists.

Obviously, j is invertible. Thus, j is a witness for $D_1 \oplus D'_1 \simeq D_2 \oplus D'_2$, where $D'_2 = j(D'_1)$. Then, because $D_1 \oplus D'_1 \simeq D_2 \oplus D''_2$, and being $\simeq$ an equivalence relation (thus transitive), $D_2 \oplus D'_2 \simeq D_2 \oplus D''_2$. Finally, $\mathcal{P}_2$ being uniform, $D_2 \rightarrow D'_2$.

It can be seen that σ_1 and σ_2 are equivalent for φ w.r.t. D'_1 and D'_2. To see that $D'_1 \approx D'_2$, observe that, since $D'_2 \simeq D''_2$ and $\mathcal{P}_2$ is uniform, by Prop. 3.8, $D'_2 \approx D''_2$. Thus, since $D'_1 \approx D''_2$ and $\approx$ is transitive, $D'_1 \approx D'_2$.

The proof for 2 is similar to 1, and is omitted. □

To state next results, we require the following notion. A *temporal epistemic run* r from $D \in \mathcal{S}$ is an infinite sequence $D^0 \rightsquigarrow D^1 \rightsquigarrow \ldots$ such that $D^i \rightarrow D^{i+1}$ or $D^i \sim_k D^{i+1}$, for some $k \in Ag$. For $n \in \mathbb{N}$, $r(n) \doteq D^n$. A state D' is said to be *t.e. reachable from* D iff there exists a temporal epistemic run r from the initial global state $r(0) = D$ such that $r(i) = D'$, for some $i \geq 0$. T.e. runs are not to be confused with the (purely temporal) runs introduced in Sec. 2.

Prop. 3.9 generalizes to temporal epistemic runs.

Proposition 3.10 *Consider two bisimilar uniform AC-MAS $\mathcal{P}_1$ and $\mathcal{P}_2$, two $\mathcal{D}$-instances $D_1 \in \mathcal{P}_1$ and $D_2 \in \mathcal{P}_2$ s.t. $D_1 \approx D_2$, an FO-CTLK formula φ, and two assignments σ_1 and σ_2 equivalent for φ w.r.t. D_1 and D_2.*

For every t.e. run r_1, if $r_1(0) = D_1$ and for all $i \geq 0$, $|U_2| \geq |ad(r_1(i)) \cup ad(r_1(i+1)) \cup C \cup \sigma_1(free(\varphi))|$, then there exists a t.e. run r_2 s.t. for all $i \geq 0$:

(i) $r_2(0) = D_2$;

(ii) $r_1(i) \approx r_2(i)$;

(iii) σ_1 *and* σ_2 *are equivalent for* φ *w.r.t.* $r_1(i)$ *and* $r_2(i)$.

Proof (sketch). The proof is an application of Prop. 3.9. □

We can now introduce the following key result.

Lemma 3.11 *Consider two bisimilar uniform AC-MAS $\mathcal{P}_1$ and $\mathcal{P}_2$, two $\mathcal{D}$-instances $D_1 \in \mathcal{P}_1$ and $D_2 \in \mathcal{P}_2$ s.t. $D_1 \approx D_2$, an FO-CTLK formula φ, and two assignments σ_1 and σ_2 equivalent for φ w.r.t D_1 and D_2. If*

1. *for every t.e. run r_1 s.t. $r_1(0) = D_1$, for all $k \geq 0$ we have $|U_2| \geq |ad(r_1(k)) \cup ad(r_1(k+1)) \cup C \cup \sigma_1(free(\varphi))| + |var(\varphi) \setminus free(\varphi)|$;*
2. *for every t.e. run r_2 s.t. $r_2(0) = D_2$, for all $k \geq 0$ we have $|U_1| \geq |ad(r_2(k)) \cup ad(r_2(k+1)) \cup C \cup \sigma_2(free(\varphi))| + |var(\varphi) \setminus free(\varphi)|$;*

then

$$(\mathcal{P}_1, D_1, \sigma_1) \models \varphi \quad \textit{iff} \quad (\mathcal{P}_2, D_2, \sigma_2) \models \varphi$$

Proof (sketch). By induction on the structure of φ, by using Prop. 3.10. We prove selected cases for the $\Rightarrow$ part. The $\Leftarrow$ part is proved analogously, by swapping indexes 1 and 2, and observing that the hypotheses are symmetric. The base case follows from Proposition 3.3. The inductive cases for propositional connectives are straightforward.

For $\varphi \equiv \forall x \psi$, assume that $x \in free(\psi)$ (otherwise consider ψ), and no variable is quantified more than once (otherwise rename variables). Let γ be a bijection witnessing that σ_1 and σ_2 are equivalent for φ w.r.t. D_1 and D_2. For each $u \in ad(D_1)$, consider the assignment $\sigma_1\binom{x}{u}$. By definition $\gamma(u) \in ad(D_2)$, and $\sigma_2\binom{x}{\gamma(u)}$ is well-defined. By noticing that $con(\psi) = con(\varphi)$ and $free(\psi) = free(\varphi) \cup \{x\}$, it is apparent that $\sigma_1\binom{x}{u}$ and $\sigma_2\binom{x}{\gamma(u)}$ are equivalent for ψ w.r.t. D_1 and D_2. Moreover, it can be seen that $|\sigma_1\binom{x}{u}(free(\psi))| \leq |\sigma_1(free(\varphi))| + 1$, as u may not occur in $\sigma_1(free(\varphi))$. Similarly for σ_2. Further, $|var(\psi) \setminus free(\psi)| = |var(\varphi) \setminus free(\varphi)| - 1$, as $var(\psi) = var(\varphi)$, $free(\psi) = free(\varphi) \cup \{x\}$, and $x \notin free(\varphi)$. This enables the application of the induction hypothesis, thus obtaining: $(\mathcal{P}_1, D_1, \sigma_1\binom{x}{u}) \models \psi$ iff $(\mathcal{P}_2, D_2, \sigma_2\binom{x}{\gamma(u)}) \models \psi$. Since γ is a bijection, the thesis easily follows.

For $\varphi \equiv K_i \psi$, we assume for contradiction that $(\mathcal{P}_1, D_1, \sigma_1) \models \varphi$ and $(\mathcal{P}_2, D_2, \sigma_2) \not\models \varphi$. Then, there exists a D'_2 s.t. $D_2 \sim_i D'_2$ and $(\mathcal{P}_2, D'_2, \sigma_2) \not\models \psi$. Then, by Prop. 3.10, there exists D'_1 s.t. $D'_1 \approx D'_2$, $D_1 \sim_i D'_1$, and σ_1 and σ_2 are equivalent for ψ w.r.t. D'_1 and D'_2. Thus, we can apply the induction hypothesis, and we obtain $(\mathcal{P}_1, D'_1, \sigma_1) \not\models \psi$. Hence $(\mathcal{P}_1, D_1, \sigma_1) \not\models K_i \psi$, which is a contradiction. □

We can now state the main result of this section.

Theorem 3.12 *Consider two bisimilar uniform AC-MAS $\mathcal{P}_1$ and $\mathcal{P}_2$, and an FO-CTLK formula φ. If*

1. *for all t.e. run r_1 s.t. $r_1(0) = D_{10}$, and for all $k \geq 0$, $|U_2| \geq |ad(r_1(k)) \cup ad(r_1(k+1)) \cup C| + |var(\varphi)|$*
2. *for all t.e. run r_2 s.t. $r_2(0) = D_{20}$, and for all $k \geq 0$, $|U_1| \geq |ad(r_2(k)) \cup ad(r_2(k+1)) \cup C| + |var(\varphi)|$*

then

$$\mathcal{P}_1 \models \varphi \quad \textit{iff} \quad \mathcal{P}_2 \models \varphi$$

Proof. (sketch) The proof follows from Lemma 3.11 by observing that hypotheses 1 and 2 imply, respectively, hypotheses 1 and 2 of Lemma 3.11. □

In this section we proved that bisimilar AC-MAS satisfying assumptions 1 and 2 in Lemma 3.12 validate the same FO-CTLK formulas. In the next section, we exploit this result to reduce, under some additional hypothesis, the verification of an infinite-state AC-MAS to that of a finite-state one.

Finite Abstraction

We now proceed to define a notion of bound in AC-MAS executions, and present abstraction results. In the rest of the paper we assume without loss of generality that any AC-MAS $\mathcal{P}$ is s.t. $ad(D_0) \subseteq C$. If this is not the case, C can be extended so as to include all the (finitely many) elements in $ad(D_0)$.

Definition 3.13 (Bounded AC-MAS) *An AC-MAS $\mathcal{P}$ is b-bounded, for $b \in \mathbb{N}$, if for all $D \in \mathcal{S}$, $|ad(D)| \leq b$.*

Observe that the notion of boundedness imposes no requirement on the interpretation domain of $\mathcal{P}$. As a consequence, even though each state in $\mathcal{P}$ does not contain more than b distinct elements, if the interpretation domain U is infinite, so is in general the state space of $\mathcal{P}$.

The aim of this section is to show that, although infinite-state, uniform b-bounded systems can in principle be verified by resorting to techniques for finite-state model checking applied to a particular AC-MAS that is a *finite abstraction* of $\mathcal{P}$.

Definition 3.14 (Finite Abstraction) *Given a b-bounded AC-MAS $\mathcal{P}_1 = \langle \mathcal{S}_1, U_1, D_{10}, \tau_1 \rangle$ with U_1 infinite, an AC-MAS $\mathcal{P}_2 = \langle \mathcal{S}_2, U_2, D_{20}, \tau_2 \rangle$ is a* finite abstraction *of $\mathcal{P}_1$, iff U_2 is finite, $|U_2| \geq 2b + |C|$, and*

(i) $D_{20} = D_{10} \in \mathcal{S}_2$;

(ii) for every $D_1 \in \mathcal{P}_1$ and $D_2 \in \mathcal{P}_2$ s.t. $D_1 \simeq D_2$ there exists $D'_1 \in \mathcal{S}_1$ s.t. $D_1 \rightarrow_1 D'_1$ iff there exists $D'_2 \in \mathcal{S}_2$ s.t. $D_2 \rightarrow_2 D'_2$ and $D_1 \oplus D'_1 \simeq D_2 \oplus D'_2$.

We can show that boundedness and uniformity are sufficient conditions for the existence of a bisimilar finite abstraction.

Lemma 3.15 *Given a b-bounded uniform AC-MAS $\mathcal{P}_1$ over an infinite U_1, we have that:*

1. *there exists a finite abstraction $\mathcal{P}_2$ of $\mathcal{P}_1$;*
2. *every finite abstraction $\mathcal{P}_2$ of $\mathcal{P}_1$ is uniform, and $\mathcal{P}_1 \approx \mathcal{P}_2$.*

Proof (sketch). For 1, build $\mathcal{P}_2$ as follows. First, let $D_{20} = D_{10}$. Next, for every $D_1, D'_1 \in \mathcal{P}_1$ s.t. $D_1 \rightarrow D'_1$, and for every $D_2 \in \mathcal{P}_2$ s.t. $D_1 \simeq D_2$, consider all witnesses ι for $D_1 \simeq D_2$. By the boundedness hypothesis and cardinality considerations, each ι can be extended to $ad(D_1) \cup ad(D'_1) \cup C$, so as to obtain a witness ι' for $D_1 \oplus D'_1 \simeq D_2 \oplus \iota'(D'_1)$. Then we define $D'_2 = \iota'(D'_1)$ and $\tau_2(D_2, \alpha_{D_2,D'_2}) = \{D'_2\}$ where α_{D_2,D'_2} is a fresh action. Thus, $D'_2 \in \mathcal{P}_2$ and $D_2 \rightarrow D'_2$. It can be easily seen that $\mathcal{P}_2$ is a finite abstraction of $\mathcal{P}_1$ by construction.

For 2, we first note that any finite abstraction $\mathcal{P}_2$ is uniform. To see this, observe that by Prop. 3.7, it is sufficient to prove req. 1 of Def. 3.6, which follows from the fact that $\mathcal{P}_1$ is itself uniform and Def. 3.14. To prove that $\mathcal{P}_1 \approx \mathcal{P}_2$, consider the relation $B \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ s.t. $\langle D_1, D_2 \rangle \in B$ iff $D_1 \simeq D_2$. Obviously $\langle D_{10}, D_{20} \rangle \in B$. Next assume that, for $D_1, D'_1 \in \mathcal{S}_1$, and for some i, $D_1 \sim_i D'_1$ and $\langle D_1, D_2 \rangle \in B$, for some $D_2 \in \mathcal{S}_2$. Thus, $D_1 \simeq D_2$. We can define D'_2 s.t. $D_1 \oplus D'_1 \simeq D_2 \oplus D'_2$. This implies that $l_i(D_2) = l_i(D'_2)$ (because $l_i(D_1) = l_i(D'_1)$), and $D'_1 \simeq D'_2$. Furthermore, we can prove that $D'_2 \in \mathcal{S}_2$; thus $D_2 \sim_i D'_2$ and $\langle D'_1, D'_2 \rangle \in B$. □

Now we can prove the final result of this section.

Theorem 3.16 *Given a b-bounded uniform AC-MAS $\mathcal{P}_1$ over an infinite U_1 and an FO-CTLK formula φ, if $\mathcal{P}_2$ is a finite abstraction of $\mathcal{P}_1$ s.t. $|U_2| \geq 2b + |C| + |var(\varphi)|$, then*

$$\mathcal{P}_1 \models \varphi \quad \textit{iff} \quad \mathcal{P}_2 \models \varphi.$$

Proof (sketch). By Lemma 3.15, $\mathcal{P}_2$ is uniform and $\mathcal{P}_1 \approx \mathcal{P}_2$. Since $\mathcal{P}_1$ is b-bounded and $|U_2| \geq 2b + |C| + |var(\varphi)|$, Theorem 3.12 applies. Thus, $\mathcal{P}_1 \models \varphi$ iff $\mathcal{P}_2 \models \varphi$. □

This result states that by using a sufficient number of elements in $\mathcal{P}_2$, i.e., by appropriately tuning the cardinality of U_2, we can in principle reduce the verification of an infinite-state AC-MAS to the verification of a finite-state one.

The theorem above does not give a procedure for the actual construction of $\mathcal{P}_2$. In the next section we introduce a class of uniform systems for which a finite-state abstraction can be easily derived.

4 Verification of Artifact System Programs

In this section we define a notion of artifact system (AS) programs, and show that their execution produces AC-MAS that are uniform. Together with the boundedness assumption, this enables us to verify them by analysing the resulting finite abstractions they generate.

Definition 4.1 (AS Program) *An* artifact system program *is a set $AS = \{\Sigma_1, \ldots, \Sigma_n\}$ of* artifact specifications *$\Sigma_i = \langle \mathcal{D}_i, D_{i0}, \Omega_i \rangle$ where:*

- *$\mathcal{D}_i$ is the* local database schema*;*
- *D_{i0} is the* local initial state*;*
- *Ω_i is the set of* local (parametric) operations*, each of the form $\omega(\vec{x}) \doteq \langle \pi(\vec{y}), \psi(\vec{z}) \rangle$ s.t.*
  - *$\omega(\vec{x})$ is the* operation signature *and $\vec{x} = \vec{y} \cup \vec{z}$ is the set of* operation parameters*;*
  - *$\pi(\vec{y})$ is the* operation precondition*, i.e., an FO-formula over $\mathcal{D}_i$;*
  - *$\psi(\vec{z})$ is the* operation postcondition*, i.e., an FO-formula over $\mathcal{D} \cup \mathcal{D}'$.*

Local database schemas were introduced in Def. 2.6. Notice that preconditions use relational symbols from the local database only, while postconditions can use any symbol from the whole $\mathcal{D}$. For an operation $\omega(\vec{x})$, we let $con(\omega) = con(\pi) \cup con(\psi)$, $var(\omega) = var(\pi) \cup var(\psi)$, and $free(\omega) = \vec{x}$. If $\vec{x} = \emptyset$, ω is said to be *ground*. An *execution* of $\omega(\vec{x})$ with *actual parameters* $\vec{u} \in U^{|\vec{x}|}$, is the *ground* operation $\omega(\vec{u}) = \langle \pi(\vec{v}), \psi(\vec{w}) \rangle$, where $\vec{v}$ (resp. $\vec{w}$) is obtained by replacing each y_i (resp. z_i) with the value occurring in $\vec{u}$ at the same position as y_i (resp. z_i) in $\vec{x}$. Such replacements make both $\pi(\vec{v})$ and $\psi(\vec{w})$ sentences. Finally, we define the set C_{AS} of all constants mentioned in AS as $\bigcup_{i=1}^{n} \big(ad(D_{i0}) \cup \bigcup_{\omega \in \Omega_i} con(\omega)\big)$.

The semantics of an AS program AS is given in terms of the AC-MAS defined by the agents that AS induces.

Definition 4.2 (Induced Agents) *Given an AS program $AS = \{\Sigma_1, \ldots, \Sigma_n\}$ for $\Sigma_i = \langle \mathcal{D}_i, D_{i0}, \Omega_i \rangle$, and an interpretation domain U, the agents $i = \langle \mathcal{D}_i, L_i, Act_i, P_i \rangle$ induced by AS over U are defined as follows:*

- $L_i \subseteq \mathcal{D}_i(U)$*;*
- $Act_i = \{\omega(\vec{u}) \mid \omega(\vec{x}) \in \Omega_i$ *and* $\vec{u} \in U^{|\vec{x}|}\}$*;*
- $\omega(\vec{u}) \in Pr_i(l_i)$ *iff* $l_i \models \pi(\vec{v})$ *for* $\omega(\vec{u}) = \langle \pi(\vec{v}), \psi(\vec{w}) \rangle$*.*

So, induced agents are obtained from Def. 2.6 by defining the set of actions as the set of ground operations given by Ω_i over U.

Once agents are defined as above, we can compose them into an AC-MAS following Def. 2.7.

Definition 4.3 (Induced AC-MAS) *Given an AS program AS, a domain U, and the set $Ag = \{1, \ldots, n\}$ of agents induced by AS over U, the AC-MAS induced by AS over U is the tuple $\mathcal{P}_{AS,U} = \langle \mathcal{S}, U, D_0, \tau \rangle$ where:*

- *$\mathcal{S}$ is defined as in Def. 2.7;*
- *$D_0 = \langle D_{10}, \ldots, D_{n0} \rangle$ is the* initial global state*;*
- *$\vec{l'} \in \tau(\vec{l}, \langle \omega_1(\vec{u}_1), \ldots, \omega_n(\vec{u}_n) \rangle)$ iff for $\omega_i(\vec{u}_i) = \langle \pi_i(\vec{v}_i), \psi_i(\vec{w}_i) \rangle$:*
  - *$\bigcup_{i \in Ag} ad(l'_i) \subseteq \bigcup_{i \in Ag} ad(l_i) \cup \vec{w}_i \cup con(\psi_i)$;*
  - *$D \oplus D' \models \psi_i(\vec{w}_i)$, where D and D' are obtained from $\vec{l}$ and $\vec{l'}$ as discussed in Sec. 2.*

In other words the AC-MAS induced by AS over U is the execution of AS over U. Observe that AS programs can be in general executed over different interpretation domains thus resulting in different AC-MAS. Moreover, according to Def. 2.7, $\tau(\vec{l}, \langle \omega_1(\vec{u}_1), \ldots, \omega_n(\vec{u}_n) \rangle)$ is defined iff $\omega_i(\vec{u}_i) \in Pr_i(l_i)$, i.e., $l_i \models \pi_i(\vec{v}_i)$, for $i \in Ag$.

In the following we assume that any relation that does not appear in the postcondition of an AS program is left unchanged in a transition, and that every AS program induces an AC-MAS whose transition relation is serial, i.e., AC-MAS states always have successors. These are basic requirements that are easily fulfilled. In Sec. 5 we present an example of one such AS program.

We have the following result on uniformity for AS programs.

Lemma 4.4 *Given an AS program AS and a domain U, every induced AC-MAS $\mathcal{P}_{AS,U}$ is uniform for C_{AS}.*

Proof. By Prop. 3.7 it is sufficient to prove uniformity for the temporal transition relation $\rightarrow$, as $ad(D_0) \subseteq C_{AS}$. To this end, consider D, D', D'' in $\mathcal{S}$ and $D''' \in \mathcal{D}(U)$ s.t. $D \oplus D' \simeq D'' \oplus D'''$ (recall that $D = \vec{l}$, and similarly D', D'', and D'''). Also, assume that $D \rightarrow D'$, i.e., there exists $\vec{\omega} = \langle \omega_1(\vec{u}_1), \ldots, \omega_n(\vec{u}_n) \rangle \in Act$ s.t. $D' \in \tau(D, \vec{\omega})$. We have to prove that $D'' \rightarrow D'''$ as well.

To do so, consider a witness ι for $D \oplus D' \simeq D'' \oplus D'''$, and extend it to an injective function ι' on $\bigcup_{i \in Ag} \vec{u}_i$. Notice that U contains enough distinct elements for ι' to exist. We can then prove that $D'' \rightarrow D'''$ as required □

The notion of satisfaction of FO-CTLK formulas in an AC-MAS can be naturally extended to AS programs.

Definition 4.5 *Given an AS program AS, a FO-CTLK formula φ, and an assignment σ, AS* over *U* satisfies *φ* under *σ, written $(AS, U, \sigma) \models \varphi$, iff $(\mathcal{P}_{AS,U}, D_0, \sigma) \models \varphi$.*

So, the model checking problem for an AS program over a domain U is defined in terms of the model checking of induced AC-MAS $\mathcal{P}_{AS,U}$.

The following result allows us to reduce the verification of AS programs inducing a b-bounded AC-MAS over an infinite U_1 to the verification of an AS program executed over a finite U_2. Let $N_{AS} = \sum_{i \in Ag} \max_{\omega(\vec{x}) \in \Omega_i} \{|\vec{x}|\}$ be the maximum number of different parameters that can occur in a joint operation defined in the program AS.

Lemma 4.6 *If the AC-MAS $\mathcal{P}_{AS,U_1}$ induced by AS over an infinite domain U_1 is b-bounded, and the finite domain U_2 is s.t. $|U_2| \geq 2b + |C_{AS}| + N_{AS}$, then the induced AC-MAS $\mathcal{P}_{AS,U_2}$ is a finite abstraction of $\mathcal{P}_{AS,U_1}$.*

Proof. Consider Def. 3.14. The following requirements are obviously fulfilled: (i) U_2 is finite; (ii) $|U_2| \geq 2b + |C_{AS}|$; and (iii) $D_{10} = D_{20}$. For the remaining requirement, consider $D_1 \in \mathcal{P}_{AS,U_1}$ and $D_2 \in \mathcal{P}_{AS,U_2}$ s.t. $D_1 \simeq D_2$, and assume that $D_1 \rightarrow D'_1$ for some $D'_1 \in \mathcal{P}_{AS,U_1}$. We show that here exists $D'_2 \in \mathcal{P}_{AS,U_2}$ s.t. $D_2 \rightarrow D'_2$ and $D_1 \oplus D'_1 \simeq D_2 \oplus D'_2$.

By definition of induced AC-MAS, if $D_1 \rightarrow D'_1$ then there is $\vec{\omega} = \langle \omega_1(\vec{u}_1), \ldots, \omega_n(\vec{u}_n) \rangle \in Act$ s.t. $D'_1 \in \tau(D_1, \vec{\omega})$. Thus, for each $\omega_i(\vec{u}_i) = \langle \pi_i(\vec{v}_i), \psi_i(\vec{w}_i) \rangle$,

- $\omega_i(\vec{u}_i) \in Pr_i(l_i)$, that is, $l_i(D_1) \models \pi_i(\vec{v}_i)$;
- $D_1 \oplus D'_1 \models \psi_i(\vec{w}_i)$.

Since $|U_2| \geq 2b + |C_{AS}| + N_{AS}$, $\sum_{i \in Ag} |\vec{u}_i| \leq N_{AS}$, and $|ad(D_1) \cup ad(D'_1)| \leq 2b$, the isomorphism ι witnessing $D_1 \simeq D_2$ can be extended to an injective function $\iota' : ad(D_1) \cup ad(D'_1) \cup C_{AS} \cup \bigcup_{i \in Ag} \vec{u}_i \mapsto U_2$. Now, let $D'_2 = \iota'(D'_1)$. By the way ι' has been defined, it can be seen that $D_1 \oplus D'_1 \simeq D_2 \oplus D'_2$. In addition, we have that $\vec{v}_i$ and $\iota'(\vec{v}_i)$ define equivalent assignments for π_i w.r.t. $l_i(D_1)$ and $l_i(D_2)$, and similarly $\vec{w}_i$ and $\iota'(\vec{w}_i)$ for ψ_i, w.r.t. $D_1 \oplus D'_1$ and $D_2 \oplus D'_2$. Thus, by Prop 3.3, $l_i(D_1) \models \pi_i(\vec{v}_i)$ iff $l_i(D_2) \models \pi_i(\iota'(\vec{v}_i))$, and $D_1 \oplus D'_1 \models \psi_i(\vec{w}_i)$ iff $D_2 \oplus D'_2 \models \psi_i(\iota'(\vec{w}_i))$. Finally, by the definition of induced AC-MAS we have that $D'_2 \in \tau_2(D_2, \vec{\omega}')$, for $\vec{\omega}' = \langle \omega_1(\iota'(\vec{u}_1)), \ldots, \omega_n(\iota'(\vec{u}_n)) \rangle$, that is, $D_2 \rightarrow D'_2$. In a similar way we can prove that, if $D_2 \rightarrow D'_2$ then $D_1 \rightarrow D'_1$. Thus, $\mathcal{P}_{AS,U_2}$ is a finite abstraction of $\mathcal{P}_{AS,U_1}$. □

Intuitively, Lemma 4.6 says that the following diagram commutes.

$$\begin{tikzcd}
AS \arrow[r, "\text{Def. 4.3}"] \arrow[dr, "\text{Def. 4.3}"'] & \mathcal{P}_{AS,U_1} \arrow[d, "\text{Def. 3.14}"] \\
 & \mathcal{P}_{AS,U_2}
\end{tikzcd}$$

The following result is a consequence of Theorem 3.16 and Lemma 4.6

Theorem 4.7 *If the AC-MAS $\mathcal{P}_{AS,U_1}$ induced by AS over an infinite U_1 is b-bounded, and the finite domain U_2 is s.t. $|U_2| \geq 2b + |C_{AS}| + \max\{N_{AS}, |var(\varphi)|\}$, then*

$$(AS, U_1) \models \varphi \quad \textit{iff} \quad (AS, U_2) \models \varphi,$$

for every FO-CTLK formula φ.

Proof (sketch). By Lemma 4.6 $\mathcal{P}_{AS,U_2}$ is a finite abstraction of $\mathcal{P}_{AS,U_1}$. Moreover, $|U_2| \geq 2b + |C_{AS}| + \max\{N_{AS}, |var(\varphi)|\}$ implies $|U_2| \geq 2b + |C_{AS}| + |var(\varphi)|$. Hence, we can apply Theorem 3.16 and the result follows. □

Thus, under the boundedness assumption, we can verify whether or not an FO-CTLK specification is satisfied by an AS program by model checking its finite abstraction.

Observe that the finite abstraction considered above depends on the specification φ. Thus, in principle, to check a different specification φ', one is required to execute AS on a different interpretation domain U_2'. However, it can be seen that this is not always required, as abstractions can usefully be re-used. Formally, let FO-CTLK$_k$ be the set of all FO-CTLK formulas containing at most k distinct variables. We have the following corollary to Theorem 4.7.

Corollary 4.8 *If $|U_2| \geq 2b + |C_{AS}| + \max\{N_{AS}, k\}$, then, for every FO-CTLK$_k$ formula φ, $(AS, U_1) \models \varphi$ iff $(AS, U_2) \models \varphi$.*

This result holds in particular for $k = N_{AS}$; thus for FO-CTLK$_{N_{AS}}$ formulas, we have an abstraction procedure that is specification-independent.

The Complexity of Model Checking FO-CTLK

We now briefly analyse the complexity of the model checking problem for finite AC-MAS w.r.t. the specification language FO-CTLK. This is tantamount to, given an AC-MAS $\mathcal{P}$ on a finite domain U and an FO-CTLK formula φ, finding an assigment σ such that $(\mathcal{P}, D_0, \sigma) \models \varphi$. Hereafter we follow (Grohe 2001) for the setting of our investigation. We encode AC-MAS by listing the elements in the domain U, the states in $\mathcal{S}$, and all the tuples of all relations. The length of the encoding of the AC-MAS $\mathcal{P}$ is denoted by $||\mathcal{P}||$. For a database schema $\mathcal{D}$ we have $||\mathcal{P}|| = \Theta(|U| + |\mathcal{S}| + \sum_{P_i \in \mathcal{D}, D \in \mathcal{S}} |D(P_i)|)$, where $f(n) = \Theta(g(n))$ means that there exist $n_0, k_1, k_2 \in \mathbb{N}$ such that $k_1 \cdot g(n) \leq f(n) \leq k_2 \cdot g(n)$ for all $n \geq n_0$. Further, the length of the encoding of φ is denoted by $||\varphi||$. We now state the following complexity result.

Theorem 4.9 *The complexity of the model checking problem for finite AC-MAS w.r.t. the language FO-CTLK is PSPACE-complete.*

Proof (sketch). This result is obtained by combining the complexity for model checking the first-order fragment of FO-CTLK and the temporal epistemic fragment. PSPACE-hardness follows by reduction to first-order model checking.

To show that the problem is in PSPACE, we briefly describe an algorithm which works in NPSPACE. Since NPSPACE = PSPACE, the result follows. Given an AC-MAS $\mathcal{P}$ and an FO-CTLK formula φ, guess an assignment σ and check if $(\mathcal{P}, D_0, \sigma) \models \varphi$. This can be done according to the structure of φ. If φ is atomic, this check can be done in PSPACE. If φ is of the form $\forall x \psi$, then we can apply the algorithm for model checking first-order (non-modal) logic, which works in PSPACE. Finally, if the main operator in φ is either a temporal or epistemic modality, then we can apply the algorithm to model check propositional CTLK, which is in P. As a result, the total complexity is in PSPACE. □

In this section we studied the model checking problem for AC-MAS. Even if it is of interest to us, we do not consider here the implicit model checking problem (Schnoebelen 2002) defined directly on AS programs. We do remark though that Theorem 4.9 is nonetheless a notable result as it shows that for AC-MAS the complexity of model checking FO-CTLK formulas is better than the complexity of checking the propositionalisation φ' of a FO-CTLK formula φ, as φ' is usually exponential in the size of φ (Hallé et al. 2007).

5 Example: the Order-to-Cash Scenario

We now briefly apply the methodology above to the *order-to-cash* artifact system, a business scenario inspired by an IBM user case (Hull et al. 2011b). This scenario includes two agents: a manufacturer m and a customer c. The customer prepares a *purchase order* (PO), i.e., a list of products the customer needs, and submits it to the manufacturer. The manufacturer can either accept it or reject it. In the former case he prepares a *work order* (WO); in the latter he notifies the customer.

We can encode the *order-to-cash* business process as an artifact system program AS, where the artifact data models are represented as database schemata, and its evolution is characterised by an appropriate set of operations.

The database schema $\mathcal{D}_i$ for each agent i is given by:

- Customer c:
  Products(prod_code, budget)
  PO(id, prod_code, offer, status)
- Manufacturer m:
  Materials(mat_code, cost)
  WO(id, po_id, price, status)

The relations *Products* and *Materials*, as well as PO and WO are self-explanatory. Notice the attribute *status*, which accounts for the evolution of each artifact. We consider the infinite set U of alphanumeric strings as the interpretation domain. *Products* and *Materials* are the only non-empty relations in the initial database instance D_0.

The operations in Ω_c and Ω_m for the customer c and manufacturer m capture the admissible actions on the underlying databases. We discuss only the operation $createWO$ for the manufacturer. Others can be done similarly. Variables v and constants c are distinguished by the font.

- $createWO(cpo\_id, price) =$
$\langle \pi(cpo\_id, price), \psi(cpo\_id, price)\rangle$, where:

- $\pi(cpo\_id, price) \equiv$
$\exists p, o\ (CPO(cpo\_id, p, o, \mathsf{prepared}) \wedge$
$\exists cost\ Materials(p, cost) \wedge cost \leq o \leq price)$

- $\psi(cpo\_id, price) \equiv$
$\exists id\ (WO'(id, cpo\_id, price, \mathsf{preparation}) \wedge$
$\forall id', c, p, s\ (WO(id', c, p, s) \rightarrow id \neq id'))$

where ϕ_k, for $k \in \mathbb{N}$, is the FO-formula defined in Section 2, which guarantees that the bound b is not violated.

The operation $createWO$ requires that the po_id is the identifier of some existing PO, and the product p appears in the *Materials* database. Its postcondition states that, upon execution, the WO contains one additional tuple with the identifier attribute univocally set to id and the attribute *status* set to preparation. Moreover, by using formulas such as ϕ_b, we can guarantee that the AS program in question is bounded and is therefore amenable to the abstraction methodology of Section 4.

We can now investigate properties of this AS program. For instance, the following formula specifies that the manufacturer m knows that each WO has to match a corresponding PO:

$$\begin{aligned}\varphi_{match} \quad = \quad & AG\, \forall po\_id(\exists id, p, s\, WO(id, po\_id, p, s) \rightarrow \\ & \rightarrow K_m \exists p', o, s' PO(po\_id, p', o, s'))\end{aligned}$$

Other specifications describing properties of the artifact-system and the agents operating in it can be similarly formalised in FO-CTLK. By the results in Section 4 we can now reduce the problem of verifying AS against φ_{match} to an instance of finite-state model checking. Suppose we are given an initial state D_0 and consider the maximum number max of parameters and the constants C_Ω in the operations in Ω_c and Ω_m. Since our AS program is bounded by b, we can consider a finite domain U' such that $D_0 \cup C_\Omega \cup con(\varphi_{match}) \subseteq U'$ and $|U'| \geq 2b + |D_0| + |C_\Omega| + |con(\varphi_{match})| + max$. Given that U' satisfies the condition of Theorem 4.7, it follows that the AS program AS over U satisfies φ_{match} if and only if AS over U' does. But the latter is a finite-state, decidable instance of model checking which can be solved through traditional techniques.

6 Conclusions and Future Work

In this paper we put forward a methodology for verifying epistemic properties of artifact-centric systems. We proposed AC-MAS, a semantics for artifact systems, on which a first-order version of CTLK can be interpreted. We observed that the model checking problem for this logic is undecidable on these systems, and proceeded to study the class of uniform systems. We showed that, under large conditions, these systems admit finite, provably bisimilar abstractions, hence satisfy exactly the same FO-CTLK formulas. While the result is of significant theoretical importance, it falls short of providing an algorithm for the construction of such finite abstraction. This is the subject of Section 4, where we give a modular description of agent programs implementing an artifact-centric system, and show that they admit finite abstractions. This is a result of significant interest to us, as it opens the way for model-checking finite models obtained from systems originally defined on infinite domains. We exemplified this technique manually on a small example in Section 5. Our current direction of work involves implementing the technique reported on top of a state-of-the-art model checker and interfacing this with GSM, a novel, declarative language for specifying artifact-systems (Hull et al. 2011a). We remark that our definition of artifact system program, notably the first-order, declarative syntax for pre- and post-conditions is deliberately aligned to GSM, to facilitate this task.

Related Work: The bounded-abstraction approach here presented is of course inspired to a large body of work in formal verification concerning data abstraction (Wolper 1986). However, none of these tackle web-services from a multi-agent perspective as here, nor, as far as we are aware, share the methodology here presented if not in general terms.

Much closer to the work presented here is (Belardinelli, Lomuscio, and Patrizi 2011b), where a data-bounded methodology for artifact-systems is put forward. Similarly to the one used here, the semantics considered by the authors is also a first-order extension of interpreted systems. However, the results presented there are considerably more limited than those in this paper. Specifically, (Belardinelli, Lomuscio, and Patrizi 2011a) only deals with a restricted version of quantified temporal logic, where no temporal modality can occur within the scope of a quantifiers. By contrast, our results here deal with the full *first-order version of CTLK* and, through bisimulations, show that, for the large classes of AC-MAS here identified, abstract and concrete models satisfy precisely the same specifications. Also differently from (Belardinelli, Lomuscio, and Patrizi 2011b), here we reason in terms of modular program description of artifact systems, which are closely related to GSM, a declarative programming language for artifact systems, and not in terms of abstract models.

An analysis of abstraction methodologies for artifact-systems was also discussed in (Belardinelli, Lomuscio, and Patrizi 2011a). However, their contribution is mostly concerned to a semantics for artifact-systems and existential abstraction results linking quotient models to the concrete ones. Differently from the present paper, no concrete methodology is put forward there to obtain finite abstractions, nor is it possible to operate on operational descriptions as the artifact-centric programs here analysed.

More broadly in the space of artifact-centric research we highlight (Deutsch et al. 2009; Cangialosi et al. 2010), which explore decidable fragments by limiting the syntax of the program descriptions of artifacts. However these approaches do not take a MAS perspective (no actors are modelled) and suffer from severe constraints in the specification language they support.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Baukus, K., and van der Meyden, R. 2004. A knowledge based analysis of cache coherence. In Davies, J.; Schulte, W.; and Barnett, M., eds., *ICFEM*, volume 3308 of *Lecture Notes in Computer Science*, 99–114. Springer.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2011a. A Computationally-Grounded Semantics for Artifact-Centric Systems and Abstraction Results. In *Proceedings of IJCAI*, 738–743. AAAI Press.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2011b. Verification of Deployed Artifact Systems via Data Abstraction. In *Proceedings of ICSOC, Paphos, Cyprus*. Spinger.

Boureanu, I.; Cohen, M.; and Lomuscio, A. 2009. A compilation method for the verification of temporal-epistemic properties of cryptographic protocols. *Journal of Applied Non-Classical Logics* 19(4):463–487.

Cangialosi, P.; G. De Giacomo; R. De Masellis; and Rosati, R. 2010. Conjunctive Artifact-Centric Services. In *ICSOC*, 318–333.

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

Cohn, D., and Hull, R. 2009. Business Artifacts: A Data-Centric Approach to Modeling Business Operations and Processes. *IEEE Data Eng. Bull.* 32(3):3–9.

Deutsch, A.; Hull, R.; Patrizi, F.; and Vianu, V. 2009. Automatic Verification of Data-centric Business Processes. In *Proc. of ICDT*.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. The MIT Press.

Gammie, P., and van der Meyden, R. 2004. MCK: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, 479–483. Springer-Verlag.

Grohe, M. 2001. Generalized model-checking problems for first-order logic. In Ferreira, A., and Reichel, H., eds., *STACS*, volume 2010 of *Lecture Notes in Computer Science*, 12–26. Springer.

Hallé, S.; Villemaire, R.; Cherkaoui, O.; and Ghandour, B. 2007. Model checking data-aware workflow properties with ctl-fo+. In *EDOC*, 267–278. IEEE Computer Society.

Hull, R.; Damaggio, E.; De Masellis, R.; Fournier, F.; Gupta, M.; Heath, III, F. T.; Hobson, S.; Linehan, M.; Maradugu, S.; Nigam, A.; Sukaviriya, P. N.; and Vaculin, R. 2011a. Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS '11, 51–62. New York, NY, USA: ACM.

Hull, R.; Damaggio, E.; De Masellis, R.; Fournier, F.; Gupta, M.; Heath III, F. T.; Hobson, S.; Linehan, M. H.; Maradugu, S.; Nigam, A.; Sukaviriya, P.; and Vaculín, R. 2011b. Business Artifacts with Guard-Stage-Milestone Lifecycles: Managing Artifact Interactions with Conditions and Events. In *Proc. of DEBS*. To appear.

Kacprzak, M.; Nabialek, W.; Niewiadomski, A.; Penczek, W.; Pólrola, A.; Szreter, M.; Wozna, B.; and Zbrzezny, A. 2008. Verics 2007 - a model checker for knowledge and real-time. *Fundamenta Informaticae* 85(1-4):313–328.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. Mcmas: A model checker for the verification of multi-agent systems. In Bouajjani, A., and Maler, O., eds., *CAV*, volume 5643 of *Lecture Notes in Computer Science*, 682–688. Springer.

Parikh, R., and Ramanujam, R. 1985. Distributed processes and the logic of knowledge. In *Logic of Programs*, 256–268.

Schnoebelen, P. 2002. The complexity of temporal logic model checking. In Balbiani, P.; Suzuki, N.; Wolter, F.; and Zakharyaschev, M., eds., *Advances in Modal Logic*, 393–436. King's College Publications.

Wolper, P. 1986. Expressing Interesting Properties of Programs in Propositional Temporal Logic. In *POPL*, 184–193.