Verification of Non-Uniform and Unbounded Artifact-Centric Systems: Decidability through Abstraction

Paper 114

ABSTRACT

The formal verification of Artifact-centric (AC) systems is a subject of growing interest in the Service Oriented Computing (SOC) community, which can benefit from techniques developed for Multi-agent systems and knowledge reasoning and representation. In the present contribution we consider the verification of AC systems that do not necessarily satisfy boundedness and uniformity, the typical assumptions used to prove decidability of the model checking problem in this setting. We provide a partial model checking procedure for agent-based AC systems against a first-order temporal logic that includes modal operators for agent knowledge. Interestingly, we obtain this result by introducing a counterpart semantics for first-order modal logic, and by defining notions of simulation and abstraction for this setting. This allows us to generate finite abstractions of infinite-state AC systems, even when these are not bounded nor uniform, thus enabling us to perform verification also in cases not covered by the current state-of-the-art.

Categories and Subject Descriptors

F.4 [Theory of computation]: Modal and temporal logics

General Terms

Languages, Theory, Verification.

Keywords

Temporal Epistemic Logic, Verification of Agent-based Systems.

1. INTRODUCTION

Artifact-centric (AC) systems have been recently put forward as a framework for the design, implementation and integration of business processes in Service Oriented Computing (SOC) [20, 21]. In the artifact paradigm the service data model and the business processes are seen as equally important components of the interface specification. This is in marked contrast with most of the traditional approaches to web-service architectures, which usually abstract data away to reduce the complexity of the system description [24]. In fact, artifacts allow for explicit dependence of state transitions on information contained in the data model. However, this enhanced expressiveness comes at a price. The presence of data means that the typical questions pertaining to system verification are much more difficult to answer. Indeed, the manipulation of data structures entails a possibly infinite state-space, which cannot be immediately handled by standard verification techniques.

In this paper we improve the state-of-the-art on artifact verification by developing a methodology to model check a class of AC systems that includes agents to account for the services operating on artifacts. Model checking is a success story on the application of formal methods in computer science [11, 2]. It has allowed for the rigorous verification of complex systems against rich specifications [10, 22, 23]. Hence, it is only natural that this technique has already been applied to the formal verification of AC systems. Since the model checking problem for AC systems in their most general setting is undecidable, the contributions in this area have focused on finding syntactic and semantics restrictions on either the system specifications or the class of relevant models, in order to ensure a decidable model checking problem, while still admitting most scenarios of interest [6, 19, 18]. Recently, a condition known as uniformity has been proved sufficient to ensure decidability of model checking when combined with a boundedness assumption on the number of active elements in each state [5, 6]. Also, uniformity was shown to be satisfied by a number of AC frameworks appearing in the literature [19, 6], thus highlighting the relevance of this condition. Therefore, a natural question is whether uniformity and boundedness are really necessary to model check AC systems. For practical purposes, the latter condition is usually difficult to check, thus more general techniques would be welcome.

In addition, most contributions on the verification of AC systems disregard the services operating on artifacts, hence modelling the system evolution in a monolithic manner. Hereafter we depart from this approach and formalize AC systems within an agent-based framework that accounts for processes operating on artifacts. We do so by relying on results on Multi-agent system (MAS) and knowledge reasoning and representation. Specifically, we model services as autonomous and proactive agents [26], and consider a specification language containing epistemic operators to express agent knowledge [17].

The main contribution of this paper can then be summarized as follows. We first introduce a formalisation of AC systems as a particular type of artifact-centric multi-agent systems (AC-MAS) [5, 6]. The latter can intuitively be seen as databases evolving in time and manipulated by agents. Differently from the references above, the AC-MAS presented hereafter are not necessarily uniform. This feature makes the verification techniques appearing in the state-of-the-art not applicable to the present framework. We state the model checking problem with respect to a first-order temporal epistemic logic including function symbols. We remark that the presence of functions increases the expressive power of our
specification language. However, it has not previously considered
in the literature to our knowledge, as AC systems on such a
language are not uniform in general. Most importantly, we
develop a novel verification methodology for non-uniform and unbounded
AC-MAS inspired to [3], and introduce a sound, albeit incomplete,
model checking procedure with respect to formulas in the universal
fragment of our first-order temporal epistemic logic. We deem the
proposed technique of interest for the model theory of first-order
modal logic (FOML) as well. Indeed, to obtain finite abstractions
of infinite AC systems we make use of the counterpart semantics
for FOML [9, 12], we define a notion of simulation for counterpart
models, and then introduce quotient structures and similar finite ab-
straction. Thus, we believe that our contribution has a theoretical
interest for modal logicians beyond the verification of AC systems.

**Scheme of the paper.** In Section 2 we fix the notation, give
the syntax of the first-order temporal epistemic logic FO-CTLK, as
well as an agent-based setting for AC systems. A counterpart se-
manitics for AC systems and a notion of simulation are presented
in Section 3. Section 4 contains the main result of the paper, i.e.,
the definition of finite abstractions for non-uniform and unbounded
AC-MAS. Section 4.1 explores constructive methods to derive ab-
stract AC-MAS. We conclude in Section 5 by discussing the results
obtained and pointing to future work. For reasons of space and sake
of presentation all proofs are omitted.

## 2. PRELIMINARIES

In this section we present the artifact-centric multi-agent sys-
tems (AC-MAS) as a multi-agent setting for AC systems [5, 6],
we introduce a first-order version of the temporal epistemic logic
CTLK, including function symbols for individuals, and state the
corresponding model checking problem. We first present the basic
terminology on databases that is used throughout the paper [1].

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

Given a (possibly infinite) interpretation domain \( U \), a \( \mathcal{D} \)-instance over \( U \) is a mapping \( D \) associating each predicate symbol \( P_i \) to a finite \( q_i \)-ary relation on \( U \), i.e., \( D(P_i) \subseteq U^{q_i} \).

The set \( \mathcal{D}(U) \) contains all the \( \mathcal{D} \)-instances on the domain \( U \). The active domain \( \text{adm}(D) \) of a \( \mathcal{D} \)-instance \( D \) is the finite set of all individuals occurring in some predicate interpretation \( D(P_i) \).

Also, the **primed version** of a database schema \( \mathcal{D} \) as above is the schema \( \mathcal{D}' = \{ P'_1/q_1, \ldots, P'_n/q_n \} \). Then, the disjoint union \( D \uplus D' \) of \( \mathcal{D} \)-instances \( D \) and \( D' \) is the \( (D \cup D') \)-instance s.t. (i) \( D \uplus D'(P_i) = D(P_i) \), and (ii) \( D \uplus D'(P'_i) = D'(P'_i) \).

We now introduce the notion of service agent, i.e., an agent op-
erating a service in an artifact-centric system.

**Definition 2 (Service agent).** Given an interpretation do-
main \( U \), a service agent is a tuple \( A = \langle \mathcal{D}, \text{Act}, \text{Pr} \rangle \), where

- \( \mathcal{D} \) is the local database schema;
- \( \text{Act} \) is the finite set of action types \( \alpha(\vec{p}) \), where \( \vec{p} \) is a tuple of abstract parameters;
- \( \text{Pr} : \mathcal{D}(U) \rightarrow 2^{\text{Act}(U)} \) is the local protocol function, where \( \text{Act}(U) \) is the set of ground actions \( \alpha \vec{u} \), for \( \alpha(\vec{p}) \in \text{Act} \), and \( \vec{u} \in U^{\vec{p}} \) a tuple of ground parameters.

Intuitively, in each moment the service agent \( A \) is in some local
state \( l \in \mathcal{D}(U) \) that represents all the information she has about
the system. In this respect we follow the typical approach to MAS
[17, 25], but here we require that this information is structured as a
database. Also, as standard we assume that agents are autonomous
and proactive, and perform the actions in \( \text{Act} \) according to the pro-
tocol function \( \text{Pr} \). As we are interested in the interactions of ser-
vice agents among themselves and with the external environment,
we define their synchronous composition.

**Definition 3 (AC-MAS).** Given an interpretation domain
\( U \) and a set \( A_g = \{ A_0, \ldots, A_n \} \) of service agents \( A_i = \langle \mathcal{D}_i, \text{Act}_i, \text{Pr}_i \rangle \) defined on \( U \), an artifact-centric multi-agent system is a tuple \( \mathcal{P} = (A_g, s_0, \tau) \) where

- \( s_0 \in \mathcal{D}(U) \) is the initial global state;
- \( \tau : \mathcal{D}_0(U) \times \ldots \times \mathcal{D}_n(U) \rightarrow \mathcal{D}(U) \) is the global transition function, where \( \text{Act}(U) = \text{Act}_0(U) \times \ldots \times \text{Act}_n(U) \) is the set of (global) actions, and

An AC-MAS evolves from the initial state \( s_0 \) according to the
global transition function \( \tau \), which returns a set of successor states
for each possible joint action of service agents. We express the de-
pendency of transitions on data by instantiating the formal param-
ters of actions with different ground individuals. Since the domain
of interpretation \( U \) is infinite in general, this means that AC-MAS
are infinite-state systems. In this respect, AC-MAS can be seen as
a natural extension of interpreted systems [17] to the first order.

We claim that the framework of AC-MAS is rich enough to for-
malise AC systems as presented, for instance, in [21, 14]. Indeed,
the data model of AC systems can be translated into the database
schema of AC-MAS, while its lifecycle can be modelled by the
transition function \( \tau \). Some of the more complex features of AC
systems, such as tasks and messages, are abstracted in AC-MAS by
the use of service agents. Nonetheless, in [4, 6] it is shown that
AC-MAS are adequate to represent the Guard-Stage-Milestone (GSM)
models for AC systems, as well as their small-step semantics.

Besides the formalisation of AC systems, we believe that AC-
MAS have a more general, theoretical interest, as these represent
relational models where each state is a finite first-order structure.
In this respect, results available for AC-MAS transfer to first-order
modal logic not dissimilarly from the way theorems on finite model
theory find applications in database theory. Thus, a study of AC-
MAS might benefit the model theory of FOML.

We now introduce some technical notions that will be used in
the rest of the paper. We denote a global ground action as \( \alpha(\vec{u}) \), where
\( \alpha = \alpha_0(p_0), \ldots, \alpha_n(p_n) \) and \( \vec{u} = (u_0, \ldots, u_n) \), and define the transition relation \( \rightarrow \) on global states such that \( s \rightarrow s' \) if there exists
\( \alpha(\vec{u}) \in \text{Act}(U) \) such that \( \alpha(\vec{u}) \rightarrow' s' \), i.e., \( s' \in \tau(s, \alpha(\vec{u})) \). A run \( \tau \) from a state \( s \) is an infinite sequence \( s^0 \rightarrow s^1 \rightarrow \ldots \), with
\( s^0 = s \). For \( n \in \mathbb{N} \), we define \( \tau(n) = s^n \). A state \( s' \) is reachable from \( s \) if there exists a run \( \tau \) from \( \tau(0) = s \) such that \( \tau(i) = s' \) for some \( i \geq 0 \). Hereafter we assume that the relation \( \rightarrow \) is serial.
This can be ensured by using skip actions. Further, we define \( S \)
the set of states reachable from the initial state \( s_0 \). As in proposi-
tional interpreted systems [17], two global states \( s = (l_0, \ldots, l_n) \)
and \( s' = (l'_0, \ldots, l'_n) \) are epistemically indistinguishable for ser-
vice agent \( A_i \), written \( s \sim_i s' \), if \( l_i = l'_i \). Differently from propositional interpreted systems, the local equality is evaluated
on database instances. Also, since we allow \( U \) to be infinite, the
set \( S \) of reachable states is also infinite in principle. Indeed,
in the general case our AC-MAS are infinite-state systems. Finally,
for technical reasons we will refer to the global database schema
\( D = D_0 \uplus \ldots \uplus D_n \) of an AC-MAS. Hence, every global state
\( s = (l_0, \ldots, l_n) \) is associated with the \( \mathcal{D} \)-instance \( D_s \subseteq D(U) \)
such that \( D_s(P_i) = \bigcup_{j \in A} D_j(l_j) \) for \( P_i \in \mathcal{D} \), that is, we as-
sume that each service agent has a truthful, yet limited, view of the
A formula \( \varphi \) is true at \( s \), written \( (P, s) \models \varphi \), if \( (P, s, \sigma) \models \varphi \) for all \( \sigma ; \varphi \) is true in \( P \), written \( P \models \varphi \), if \( (P, s_0) \models \varphi \).

Notice that we adopt an active domain semantics, where quantifiers range over the active domain \( \text{dom}(s) \) of \( s \). This is a standard assumption in database theory.

Finally, we present the model checking problem for AC-MAS with respect to the specification language FO-CTLK.

**Definition 6 (Model Checking Problem).** Given an AC-MAS \( P \) and an FO-CTLK formula \( \varphi \), determine whether there is an assignment \( \sigma_0 \) such that \( (P, s_0, \sigma_0) \models \varphi \).

Model checking general AC-MAS is known to be undecidable. In [5] this problem is proved to be decidable for bounded and uniform systems. We now introduce both notions in relation to AC-MAS on a language with only variables and constants as individual terms, which is the original setting of [5]. We first define a notion of isomorphism between states: two states \( s \) and \( s' \) are isomorphic, or \( s \simeq s' \), if there exists a bijection \( \iota : \text{dom}(s) \cup \text{dom}(s') \to \text{dom}(s) \cup \text{dom}(s') \) s.t. \( i(s) \) is the identity on \( \text{dom}(s) \) and \( (ii) \) for every \( i \in \text{dom}(s), P_i \in P, u \in \mathbb{U}, \iota(i(P_i)) = \iota(i(P')). \)

**Definition 7 (Boundedness and Uniformity).** An AC-MAS \( P \) is bounded iff there exists \( b \in \mathbb{N} \) such that for all \( s \in \text{dom}(s) \leq b \).

An AC-MAS \( P \) is uniform iff for \( s, s' \in S, t' \in D(U), \) if \( s \alpha(s,t) \equiv s \\
\alpha(s,t') \equiv t' \) for every constant-preserving bijection \( i \) extending \( \iota \).

Notice that the boundedness condition restricts the number of elements appearing in the active domain of each state, not the total number of states in \( S \), which is infinite in general. In [5] the combination of both features is proved sufficient to obtain a decidable model checking problem. Yet, in [3] boundedness is shown not strong enough to ensure decidability alone. To conclude, we illustrate the formal machinery introduced thus far with a running example.

**Example 1.** While the following example is relatively small and designed on purpose, it is nonetheless instructive to describe the relevant features of AC-MAS introduced above. It elaborates on a similar example appeared in [3]. Assume \( \mathbb{N} \) as the interpretation domain and consider a set \( A_0 = \{A_0, A_1, A_2\} \) of service agents \( A_i = \{D_i, Act_i, Pr_i\} \) defined as follows: for \( i \leq 2 \), \( D_i = \{P_1, \beta\} \); \( ii \) \( Act_i = \{\alpha_i(n)\} \); and \( (iii) \) for every \( l_i \in D_i(N), Pr_i(l_i) = \{\alpha_i(n)\} \). Now let \( P_0 = \{Ag, s_0, \tau\} \) be the AC-MAS depicted in Fig. 1(a), where

- the initial state \( s_0 \) is equal to \( (\langle P(0)\rangle, \emptyset, \emptyset) \);
- \( s' = \tau(s, \alpha(n)) \) whenever exactly one of the \( l_i \) contains \( P(n) \), while \( l_{i+1+\mathcal{M}} \langle x \rangle \) contains exactly \( P(n + 1) \). Any other local state is empty.

Notice that for every state \( s' \) and \( n \in \mathbb{N} \) there exists at most one \( s \) such that \( s \alpha(n) \equiv s' \). Moreover, \( P_0 \) is non-uniform. Indeed, \( s_0 \alpha_0 \sim s_0\alpha_0 \equiv s_4 \) with bijection \( \iota(0) = 0 \) and \( \iota(1) = 4 \). Also, \( s_0 \alpha_0 \sim s_4 \), but it is not the case that \( s_0 \alpha_0 \sim s_4 \).

Furthermore, as an example of an unbounded AC-MAS we consider the following modification of \( P_0 \). Define \( A_i' = \{D_i', Act_i', Pr_i'\} \) s.t. \( (i) D_i' = \{P_1, Q_2\} \); \( ii \) \( Act_i = \{\alpha'_i(n)\} \); and \( (iii) \) for every \( l_i \in D_i'(\mathbb{N}), Pr_i'(l_i) = \{\alpha'_i(n)\} \). For \( A'_0 = \{A_0, A_1, A_2\} \) let \( P_0' = \{Ag', s'_0, \tau'\} \) be the AC-MAS depicted in Fig. 1(b), where

- the initial state \( s'_0 \) is equal to \( s_0 = (\langle P(0)\rangle, \emptyset, \emptyset) \)
Counterpart semantics was originally conceived as a semantics for first-order modal logic in which we do not need to assume that the same individual appears in more than one system state. In fact, the task of identifying “related” individuals across states is performed by the counterpart relations.

Similarly to AC-MAS, the active domain adom(s) of a state s is defined as the set of all individuals occurring in some predicate interpretation I(P^n, s). We remark without proof that any AC-MAS can be seen as an MA c-model on a finite language, where the counterpart relation is the identity, and constants and functions are interpreted as themselves.

**Remark 1.** Let \( \mathcal{P} = (\mathcal{A}, s_0, \tau) \) be an AC-MAS on the domain \( U \). Consider \( \mathcal{M}_\mathcal{P}^{\tau} = (\mathcal{S}, s_0, \mathcal{U}^\tau, \rightarrow, C^\tau, \{\sim_i\}_{i \in \mathcal{A}_\mathcal{P}}, \{C^n_i\}_{i \in \mathcal{A}_\mathcal{P}}, I) \), where \( (i) \mathcal{S} \) is the set of reachable states in \( \mathcal{P} \); (ii) \( s_0 \) is the initial state in \( \mathcal{P} \); (iii) \( \rightarrow \) and each \( \sim_i \) are defined as in Section 2; and for all \( s, s' \in \mathcal{S} \), (iv) \( \mathcal{U}^\tau(s) = U \); (v) \( C^\tau_i \) and each \( C^n_i \) are the identity relation; (vi) \( I(c, s) = c \). If \( f^k \in F \), and \( I(P^n, s) = D_s(P^n) \). Then \( \mathcal{M}_\mathcal{P}^{\tau} \) is an MA c-model.

The notion of run is defined as for AC-MAS. The relation \( C^{\tau+} \) is the transitive closure of \( C^\tau \), i.e., \( C^{\tau+}_i(a, a') \) iff there is a sequence \( s^0 \rightarrow s^1 \rightarrow \ldots \rightarrow s^k \) s.t. \( s^0 = s \), \( s^k = s' \), and there are \( a^0, \ldots, a^k \) s.t. \( a^0 = a, a^k = a' \), and \( C^{\tau+}_i(a^i, a^{i+1}) \) for \( i < k \).

As in Section 2, \( \sim \) is the transitive closure of \( \cup_{A \in \mathcal{A}_\mathcal{P}} \sim_i \), while for \( s, s' \in S \), \( s \sim s' \) \( C^{\tau+}_i(a, a') \) iff there is a sequence \( s^0 \sim_i s^1 \sim_i \ldots \sim_i s^k \) s.t. \( s^0 = s, s^k = s' \), and there are \( a^0, \ldots, a^k \) s.t. \( a^0 = a, a^k = a' \), and \( C^{h+1}_i(a^i, a^{i+1}) \) for \( h < k \).

To define the satisfaction of FO-CTLK in formulae of AC-MAS, we consider typed languages and finitary assignments. This is standard when working with counterpart semantics [9, 12]. Specifically, every variable \( x_i \in \mathcal{V} \) is a term of type \( n \), or \( n \)-term, for \( n \geq 2 \); every constant \( c \in \mathcal{C} \) is an \( n \)-term; and if \( f^k \) is a function symbol and \( \bar{t} \) is a \( k \)-tuple of \( n \)-terms, then \( f^k(\bar{t}) \) is an \( n \)-term.

**Definition 9 (FO-CTLK^T).** The typed language FO-CTLK^T contains all \( n \)-formulas \( \phi \), for \( n \in \mathbb{N} \), as follows:

- if \( P^n \) is an \( m \)-ary predicate symbol and \( \bar{t} \) is an \( m \)-tuple of \( n \)-terms, then \( P^n(\bar{t}) \) is an (atomic) \( n \)-formula;
- if \( \psi, \psi' \) are \( n \)-formulas, then \( \neg \psi \rightarrow \psi' \), \( A \psi \), \( A \psi U \psi' \), \( E \psi U \psi' \), \( K_i \psi \) and \( C \psi \) are \( n \)-formulas;
- if \( \psi \) is an \( (n+1) \)-formula, then \( \forall x_{n+1} \psi \) is an \( n \)-formula.

The other logical operators are defined as standard. In what follows we consider also the sublanguage FO-AC-CTLK^T, which is standardly obtained by restricting FO-CTLK^T to the universal modalities \( A \), \( U \), \( K \), and \( C \). Also, the typed first-order logic FO\(_T\) is the non-modal fragment of FO-CTLK^T.

The meaning of a typed formula \( \phi : n \) at a state \( s \) can intuitively be understood as a subset of \( U(s)^n \), i.e., the set of \( n \)-tuples satisfying \( \phi : n \) at \( s \). Therefore, the definition of satisfaction is given by means of finitary assignments, where an \( n \)-assignment in \( s \) is an \( n \)-tuple \( \bar{a} \) of elements in \( U(s) \). Let \( t \) be an \( n \)-term, the valuation \( \bar{a}(t) \) for the \( n \)-assignment \( \bar{a} \) is equal to \( a_i \) if \( t = x_i \). Also, \( \bar{a}(t) = I(c, s) \) whenever \( t = c \), and \( \bar{a}(f^k(t)) = I(f^k, s)(\bar{a}(t_1), \ldots, \bar{a}(t_k)) \).

**Definition 10 (Semantics of FO-CTLK^T).** The satisfaction relation \( \models \) for a state \( s \in \mathcal{M} \), a typed formula \( \phi : n \) and an \( n \)-assignment \( \bar{a} \) is inductively defined as follows:

- if \( c \in \mathcal{C} \), then \( I(c, s) \in U(s) \); and (iii) if \( f^k \in F \), then \( I(f^k, s) \) is a function from \( U(s)^k \) to \( U(s) \).
Definition 11 (State Simulation). A state \( s' \) simulates \( s \), or \( s \preceq s' \), if there exists a surjective function \( \psi : U(s) \to U'(s') \) s.t. (i) for every constant \( c, \psi(I(c,s)) = I'(c,s') \); (ii) for every function \( f^k, \bar{u} \in U(s)^k, I'(f^k,\bar{s})(\bar{u}) = I'(f^k,\bar{s})(\bar{u}) \); (iii) and for every \( P_j \in \mathcal{D} \) and \( \bar{u} \in U(s)^{\bar{n}}, I'(P_j,\bar{s}) \bar{u} \) iff \( I(P_j,\bar{s}) \bar{u} \).

Any function \( \psi \) as above is a witness for \( s \preceq s' \). We write \( s \preceq s' \) to state this explicitly. Witnesses preserve the interpretation of terms and predicates, but not necessarily the multiplicity of individuals. Notice that by definition \( u \in \text{adom}(s) \) iff \( \psi(u) \in \text{adom}(s') \).

Definition 12 (Assignment Simulation). Let \( \bar{a} \in U(s)^n \) and \( \bar{a}' \in U'(s'^n) \) be n-assignments, \( \bar{a}' \preceq \bar{a} \) simulates \( \bar{a} \), or \( \bar{a} \preceq \bar{a}' \), iff for some witness \( \psi \), \( \bar{a} \preceq s' \) and \( \psi(\bar{a}) = \bar{a}' \).

We overload the symbol \( \preceq \) to represent state and assignment simulations; the difference will be clear from the context. Notice that \( \preceq \) is a transitive relation on \( S \) and \( \bigcup_{s \in S} U(s) \) respectively. Also, assignment simulation preserves the interpretation of FO\( _{\bar{u}} \)-formulas.

Lemma 2. If \( (s,\bar{a}) \preceq (s',\bar{a}') \), then for every \( n \)-term \( t \) and \( n \)-formula \( \phi \) in \( \mathcal{F} \),

\[
\psi(\bar{a}(t)) = \bar{a}'(t) \quad \text{iff} \quad (M(s,\bar{a}) \models \phi) \Rightarrow (M(s',\bar{a}') \models \phi)
\]

The proof is by induction on the length of \( t \) and the length and type of \( \phi \). We now introduce the notion of simulation on MA c-models, which will be used to extend Lem. 2 to \( \mathcal{F}_{\text{ACTLK}} \).

Definition 13 (Model Simulation). The MA c-model \( M' \) simulates \( M \) or \( M \preceq M' \), iff \( (i) s_0 \preceq s'_0 \); (ii) if \( (s,\bar{a}) \preceq (s',\bar{a}') \) then for every \( t \in S \), \( \bar{b} \in U(t)^n \), if \( s \to t \) and \( C_{s,t}(\bar{a},\bar{b}) \), then there are \( t' \in S' \), \( \bar{b}' \in U'(t')^n \) s.t. \( s \to t' \), \( C_{s',t'}(\bar{a}',\bar{b}') \), and \( (t',\bar{b}') \preceq (t',\bar{b}) \); and (iii) if \( (s,\bar{a}) \preceq (s',\bar{a}') \), then for every \( t \in S' \), \( \bar{b} \in U(t'^n) \) s.t. \( s' \to t \) and \( C_{s',t}(\bar{a},\bar{b}) \), then there are \( t' \in S' \), \( \bar{b}' \in U'(t')^n \) s.t. \( s' \to t' \), \( C_{s',t'}(\bar{a}',\bar{b}') \), and \( (t',\bar{b}') \preceq (t',\bar{b}) \).

Again, we use the symbol \( \preceq \) to express a simulation between MA c-models; the difference will be clear from the context. The simulation relation \( \preceq \) extends to MA c-models the commutativity conditions of standard model simulations [8]. Most importantly, since by Remark 1 AC-MAS can be seen as a specific class of MA c-models, Def. 13 applies also to the former. Finally, we can state the main result of this section, namely, the simulation relation on MA c-models preserves the satisfaction of \( \mathcal{F}_{\text{ACTLK}} \)-formulas.

Theorem 3. Suppose that \( M \preceq M' \) and \( (s,\bar{a}) \preceq (s',\bar{a}') \). Then for every \( n \)-formula \( \phi \) in \( \mathcal{F}_{\text{ACTLK}} \),

\[
(M',s',\bar{a}') \models \phi \Rightarrow (M,s,\bar{a}) \models \phi
\]

From Thm. 3 we immediately obtain the following result.

Corollary 4. If \( M \preceq M' \) then for every \( n \)-formula \( \phi \) in \( \mathcal{F}_{\text{ACTLK}} \), \( \bar{a}_0 \in U'_{\bar{n}}(s'^n) \), there exists \( \bar{a}_0 \in U(s)^n \) such that

\[
(M',s',\bar{a}_0) \models \phi \Rightarrow (M,s,\bar{a}_0) \models \phi
\]

By Lem. 1 and Cor. 4 we can tackle the model checking problem for an AC-MAS \( P \) and an FO\( _{\bar{u}} \)-CFL formula \( \phi \) by considering an MA c-model \( M' \) that is similar to \( M \). By Cor. 4, if \( (M',s',\bar{a}_0) \models \phi \) for some n-assignment \( \bar{a}_0 \), then there exists \( \bar{a}_0 \in U(s)^n \) s.t. \( (M,s,\bar{a}_0) \models \phi \). Moreover, by Lem. 1, if \( (M',s',\bar{a}_0) \models \phi \) then \( (M',s',\bar{a}_0) \models \phi \). Therefore, the solution to the model checking problem for the abstract MA c-model \( M' \) implies a positive solution also for the concrete AC-MAS \( P \). In the following sections we analyse the conditions under which the abstract MA c-model \( M' \) is finite and can be constructively given.
4. FINITE ABSTRACTION

In this section we introduce the abstraction of a MA c-model $M$ and show that it is similar to $M$. Further, we identify the conditions under which such abstraction is finite, thus allowing the verification of infinite-state AC-MAS by the results in Section 3. Hereafter we assume that the sets $C_{i,j}$ of constants and $F$ of functions are finite. This can be done without loss of generality, as we can take $C_{i,j}$ and $F$ as the sets of constants and functions appearing in the formula to be verified. All the results in previous sections still hold for FO-CTLK formulas on this finite language.

First, we define $[s]$ as the equivalence class of $s \in S$ according to the symmetric closure $\equiv$ of the state simulation relation $\leq$. Further, for $s \in \{[i], [s, a]_{[i]}\}$ the equivalence class of $(s, a)$ according to the symmetric closure $\equiv$ of the assignment simulation relation $\leq$. Notice that we overload the symbol $\equiv$ as we did with $\leq$; the distinction will be clear from the context. Also, $\equiv$ is not to be confused with the isomorphism relation $\equiv$ in Section 2.

**Definition 14 (Abstraction).** Given an MA c-model $M = (S, s_0, U, \rightarrow, C, I^c)$, the abstraction of $M$ is a finite state abstraction $M' = (S', s_0', U', \rightarrow', C', I^c)$ such that for all $s \in S$, $s' \in S'$, $a \in A$, $s \leq s'$, $s \rightarrow a \models I^c(s)$ implies $s' \rightarrow' a \models I^c(s')$.

**Theorem 7.** For every AC-MAS $P$, the abstraction $M'$ of the MA c-model $M_0$ is finite.

As a consequence of Cor. 6 and Thm. 7, to verify an FO-CTLK formula $\phi$ on an infinite-state AC-MAS $P$, we can model check $\pi\phi(\sigma)$ in FO-CTLK$_T$ on the finite abstraction $M'$. We illustrate this methodology by elaborating on Example 1.

**Example 2.** We define the abstraction $M'_0$ of the AC-MAS $P_0$ in Example 1. The abstraction $M'_0$ of $P_1$ can be obtained similarly and it is depicted in Fig. 2(c), a detailed description is omitted for reasons of space.

We observe that $P_2$ contains a unique run $s_0 \rightarrow s_1 \rightarrow \ldots$, as depicted in Fig. 1(b). Further, for every $i > 0$, there is a surjective function $i : U \rightarrow U$ from $s_{i+1}$ to $s_i$ that satisfies the conditions in Def. 11. Specifically, the witness $i$ maps $i + 1$ to $i$, any $k < i + 1$ to some $k' < i$, and any $k > i + 1$ to some $k' > i$. As a result, for every $i > 0$, $s_{i+1} \leq s_i$. Thus, we define the set of states in the abstraction $M'_2$ as $S'_0 = \{s'_0, s'_1\}$, where $s'_0 = \{s_0\}$ and $s'_1 = \{s_i\}$ for $i > 0$, corresponding to the equivalence relation $\equiv$. In particular, $s'_0 \rightarrow s'_1$ and $s'_1 \rightarrow s'_0$ by definition of $\rightarrow'$. Further, for every $n$, $m > 0$, $(s_0, m) \leq (s_n, n)$ by the witness $i$ defined as above. Moreover, if $n' < n$, $m' < m$ (resp. $n' > n$, $m' > m$), we have that $(s_0, m') \leq (s_n, n')$. Further, and $(s_0, m) \leq (s_n, n)$. Hence, we obtain two equivalence classes on $N$ for $n = 0$ and $n > 0$: $a_0 = \{s_0\}$, $a_0 = \{s_n\}$, $\forall n' < n$ and $\forall n' > n$. Further, the counterpart relation is defined as $C^T_{s_0, s_0} = \{(a_0, c_1), (b_0, c_1), (b_0, a_0)\}$, corresponding intuitively to the transitions $(s_0, 0) \rightarrow (s_1, 0), (s_0, n') \rightarrow (s_1, n')$ for $n' > 0$, and $(s_0, 0) \rightarrow (s_1, 1)$. Further, $C^T_{s_0, s_0} = \{(a_1, c_1), (b_1, c_1), (b_1, a_1)\}$, which corresponds to the transitions $(s_n, n) \rightarrow (s_{n+1}, n), (s_{n+1}, n) \rightarrow (s_{n+1}, n+1)$, and $(s_{n+1}, n) \rightarrow (s_{n+1}, n')$ for $n' < n$. As regards the epistemic components, by Def. 14 we have that $\sim_0$ and $\sim_1$ are the identity relation, while $\sim_0$ is equal to $S' \rightarrow S'$. Moreover, $C^T_{s_0, s_0} = \{(a_0, c_1), (c_1, a_0), (b_0, b_1), (b_1, b_0), (b_0, a_0)\}$, $C^T_{s_0, s_0} = \{(a_1, c_1), (c_1, a_0), (b_0, b_1), (b_1, b_0), (b_0, a_0)\}$. Finally, we remark that the FO-CTLK formulas $\chi$ and $\Theta$ do not contain neither constants nor functions. Hence, the interpretation $I'$ is s.t. $I'(P, s'_0) = \{a_0\}, I'(Q, s'_0) = \emptyset$, $I'(P, s'_1) = \{a_1\}$, and $I'(Q, s'_1) = \{c_1, a_1\}$. The abstract MA c-model $M'_0$ is illustrated in Fig. 2(d).

We can now model check on $M'_0$ and $M'_2$ the typed versions of
\( \chi \) and \( \theta \), namely

\[
\chi_T = \forall x T K_0(P(x_1) \rightarrow AX \neg P(x_1))
\]

\[
\theta_T = \forall x T K_1(P(x_1) \rightarrow AXA \neg GP(x_1))
\]

It can be checked that \( M'_2 \models \chi_T \land \theta_T \), and \( M'_2 \models \chi_T \), while \( M'_1 \models (\chi \land \theta) \). Hence, by Lem. 1 and Cor. 6 we can derive that \( P_2 \models \chi \land \theta \), and \( P_1 \models \chi \); while nothing can be said about \( \theta \) in \( P_1 \). As a result, whenever the abstraction satisfies the FO-ACTLK specification, we are able to model check AC-MAS even though these are neither uniform nor bounded.

We conclude by remarking that, since we consider surjective mappings between states, multiple individuals can be “compressed” into one single abstract individual. Hence, the abstraction of an AC-MAS as defined in Def. 14 is not an AC-MAS in general, rather an MA c-model that includes temporal and epistemic counterpart relations on its elements. This motivates the introduction of MA c-models in first place.

### 4.1 Towards Constructive Abstraction

In the previous section we defined the abstraction of an MA c-model \( M \) and proved that it is similar to \( M \). Also, in Thm. 7 we remarked that this abstraction is finite, thus allowing for model checking a specification on a concrete, infinite-state AC-MAS \( P \) by verifying the finite abstraction of \( M_\mathcal{F} \). However, the definition of abstraction provided in Section 4 is not constructive in general, as it relies on the quotient structure \( S/\approx \) of the infinite state space \( S \) of \( M \). In this section we explore a mechanism to build finite abstractions constructively, starting from the way actions and transitions are specified. The following definition of actions has first appeared in a different form in [5].

**Definition 15 (Actions).** For each service agent \( A_i \), \( Act_i \) is the set of action types, \( \alpha(\vec{z}) = (\pi(\vec{y}), \psi(\vec{z})) \), where

- \( \pi(\vec{y}) \) is the action signature, \( \vec{z} = \vec{y} \cup \vec{z} \) as parameters;
- \( \psi(\vec{z}) \) is the action precondition, i.e., an FO\(^*\)-formula over \( D_i \);
- \( \psi(\vec{z}) \) is the action postcondition, i.e., an FO\(^*\)-formula over \( D \cup D'_i \).

Observe that we admit the use of identities in the action specifications and, differently from [5], of functions as well. Given actions as in Def. 15, we can define the transition function as follows, where \( F_\mathcal{P} \) is the finite set of function symbols appearing in a formula \( \phi \), and if \( k \) is the maximum nesting of function symbols in \( \phi \), \( F_\mathcal{P}^{\leq k} \) is the finite set of elements obtained by applying the functions \( f \in F_\mathcal{P} \) to individuals in \( \vec{u} \) at most \( k \) times.

**Definition 16 (Protocol and Transition Function).** Let \( \alpha_i(\vec{z}) = (\pi_i(\vec{y}), \psi_i(\vec{z})) \) be an action for service agent \( A_i \in \mathcal{A} \), and \( \vec{u}_i = \vec{y} \cup \vec{u}_i \) individuals in the domain \( U \). Then, \( \alpha_i(\vec{u}_i) \in P_{\mathcal{R}_i}(l_i) \iff (l_i, \sigma) \models \pi_i(\vec{y}) \land \psi_i(\vec{z}) \).

Further, for an AC-MAS \( P \) on \( Ag \) the transition function \( \tau \) is defined so that, for \( s = \langle l_0, \ldots, l_n \rangle \) and \( s' = \langle l'_0, \ldots, l'_n \rangle \) in \( \tau(s, (\alpha_i(\vec{u}_i))) \) iff for every \( A_i \in \mathcal{A} \),

1. \( (D_i \oplus l_i', \sigma') \models \psi_i(\vec{z}) \) for \( \sigma'(\vec{z}) = \vec{u}_i \)
2. \( adom(l_i') \subseteq adom(l_i) \cup F_\mathcal{P}^{\leq k} \)(\( \vec{u}_i \) \cup \text{con}(\psi_i))

In Def. 16 we consider a satisfaction relation \( \models \) between database instances and FO\(^*\)-formulas, which can be derived from Def. 5. Also, \( \oplus \) is the disjoint union of db instances introduced in Section 2. Notice that by condition (2) the number of candidates for each \( l'_i \) is finite. Also, condition (1) can be effectively computed. Hence, the transition function is decidable. Notice that in general we have multiple successors \( s' \) for given \( s \) and \( \alpha_i(\vec{u}) \), but always in finite number.

**Example 3.** We show that the agent services in Example 1 and 2 can be described by means of actions as specified in Def. 15. For each agent \( A_i \), \( i \leq 2 \), we assumed that \( Act_i = \{ \alpha_i(n) \} \), where each \( \alpha_i(n) = (\pi_i, \psi_i(n)) \) is specified as

\[
\pi_i = \text{true}
\]

\[
\psi_i = ((n-i) \% 3 = 2) \land P(n) \land P'(n+1) \land
\forall x(x \neq n + 1) \rightarrow P'(x)) \lor
((n-i) \% 3 \neq 2) \land P(n) \land \forall x \neg P'(x))
\]

Further, for \( A'_2 \) we have that \( \alpha'_2(n) = (\pi'_2, \psi'_2(n)) \) is specified as

\[
\pi'_2 = \text{true}
\]

\[
\psi'_2 = \psi_2 \land Q'(n, n+1) \land
\forall x, y(Q(x, y) \rightarrow Q'(x, n+1) \land Q'(y, n+1)) \land
(Q'(x, y) \rightarrow (y = n+1) \land \exists z(Q(x, z) \lor Q(z, x)))
\]

It can be checked that the protocols and transition function as defined in Def. 16 correspond indeed to those for the AC-MAS \( P_1 \) and \( P_2 \) in Example 1. Notice the use of function symbols ‘+1’ (successor) and ‘\% 3’ (remainder of division by 3), which entails that the AC-MAS thus obtained are not uniform in general.

Actions specified as in Def. 15 generate AC-MAS that are not necessarily uniform. This is in contrast with the situation for AC-MAS described by actions on a language without function symbols. For that restricted language it was shown that the corresponding AC-MAS are always uniform [5, 6]. Thus, the increased expressiveness of the specification language for actions requires the novel verification techniques put forward in this paper.

Now we explore methods to obtain finite abstractions constructively and consider the following condition on AC-MAS.

**Definition 17 (Weak Uniformity).** An AC-MAS \( P \) is weakly uniform iff for every \( s, s' \in S \), if \( s \approx s' \), \( s \rightarrow^\alpha(\vec{u}) \) and \( s' \rightarrow^\alpha(\vec{u'}) \), then \( t \approx t' \).

If an AC-MAS is weakly uniform, then its temporal evolution modulo the relation \( \approx \) is captured by the actions independently from ground parameters. Weak uniformity still allows for great expressiveness. In particular, the AC-MAS in Examples 1-3 are weakly uniform. Notice that weak uniformity is strictly weaker than full-fledge uniformity; for instance the AC-MAS \( P_1 \) is weakly uniform but not uniform. Also, weakly uniformity admits unboundedness as it is the case for the AC-MAS \( P_2 \).

Now we briefly describe the construction of the abstract MA c-model \( M'_\mathcal{F} \) for a weakly uniform AC-MAS \( P \). We keep this discussion informal for reasons of space. Starting from the initial state \( s_0 \), for each \( s \in S \) and each joint action \( \alpha(\vec{z}) \) we consider all tuples \( \vec{u} \) containing elements in \( adom(s) \) together with at most \( |\vec{z}| \) elements not in \( adom(s) \) (notice that the elements in \( U \setminus adom(s) \) behave in the same way w.r.t. the satisfaction of FO\(^*\)-formulas).

Obviously their number is finite. Further, if \( s \rightarrow^\alpha(\vec{u}) t \) for some \( \vec{u} \), then we add the representative \( t \) to the set of reachable states in the abstraction, unless \( t \approx t' \) for some \( t' \) already appearing in the set of reachable states. Since the number of equivalence classes according to \( \approx \) is finite, this construction terminates after a finite number of steps, returning the abstraction \( M'_\mathcal{F} \).

**Example 4.** To give a hint of the methodology proposed, we build part of the abstract MA c-model \( M'_\mathcal{F} \) for the unbounded and weakly uniform AC-MAS \( P_2 \), starting from the initial state \( s_0 \). We first observe that, since the active domain of \( s_0 \) is finite and all non-active individuals can be mapped to any \( u \in N \setminus adom(s_0) \), we obtain two equivalence classes \( s'_0 = [s_0] \), that is, \( s_0 = [s_0, 0]_{[s_0]} = \ldots \)
\{ (s_0,0) \} and \( b_0 = [s_0, n]_{[s_0]} = \{ (s_0, n) \mid n \neq 0 \} \). Then, we consider the only joint action \( \alpha(n) \) in \( \mathcal{P}' \) with \( n \in \text{dom}(\{s_0\}) = \emptyset \) or \( n \) equal to any \( u \in \mathbb{N} \setminus \text{dom}(\{s_0\}) \). The transition triggers only in the former case and we have \( s_0 \xrightarrow{\alpha(n)} s_1 \). We add \( s_1 \) to the set of reachable states in \( \mathcal{M}_2' \) as \( s_1 \neq s_0 \). Again, we consider the joint action \( \alpha(n) \) for \( n \in \text{dom}(s_1) = \{ 0, 1 \} \) or \( n \) equal to any \( u \in \mathbb{N} \setminus \text{dom}(s_1) \). Now the transition is triggered only for \( n = 1 \) with \( s_1 \xrightarrow{\alpha(1)} s_2 \). Moreover, by considering the finitely many subject functions from \( \text{dom}(s_2) \) to \( \text{dom}(s_1) \) we can check that \( s_2 \leq s_1 \), that is, \( s_2 \in [s_1]_1 \). Therefore, in the abstract \( \mathcal{M}_2' \) we only add a reflexive transition on \( [s_1]_1 \). This concludes the construction of the abstraction \( \mathcal{M}_2' \) for \( \mathcal{P}' \), as by weak uniformity it is a consequence of the fact that \( s_2 \approx s_1 \), i.e., \( s_2 \approx s_1 \), if \( s_2 \approx s_1 \), then in particular, since \( s_1 \approx s_2 \), we can check that \( \alpha(1) \approx s_2 \) and \( s_1 \approx s_2 \), it is the case that \( s_1 \approx s_2 \), i.e., \( s_1 \approx s_2 \).}

5. CONCLUSIONS AND FURTHER WORK

In this paper we introduced an abstraction methodology for a class of artifact-centric systems that was not yet tractable by the current state-of-the-art, namely non-uniform and unbounded AC-MAS. We remarked that non-uniform AC-MAS rise naturally when considering specification languages for agent actions that contain functions. This feature is often crucial to express functional dependencies between data in our underlying databases, as well as their relatedness. There is a number of directions to extend the results here presented. In particular, we aim at developing the methodology for building abstractions of weakly uniform systems outlined in Section 4.1 into an effective verification technique. In this respect, results on the complexity of the procedure, that has not been addressed here, are of utmost importance.

6. REFERENCES