A Computationally-Grounded Semantics for Artifact-Centric Systems and Abstraction Results

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
Department of Computing
Imperial College London
f.patrizi@imperial.ac.uk

Abstract

We present a formal investigation of artifact-based systems, a relatively novel framework in service oriented computing, aimed at laying the foundations for verifying these systems through model checking. We present an infinite-state, computationally grounded semantics for these systems that allows us to reason about temporal-epistemic specifications. We present abstraction techniques for the semantics that guarantee transfer of satisfaction from the abstract system to the concrete one.

1 Introduction

Considerable research in service-oriented computing, including in multi-agent systems approaches to it, has focused on tackling the orchestration and choreography problems [Alonso *et al.*, 2004]. In a nutshell these involve offering a methodology for the seamless interaction of autonomous and distributed services thereby satisfying individual and overall goals in the system. Some of the most compelling contributions to both problems have focused on the use of verification technology. In these approaches orchestration is tackled by means of synthesis, whereas "vanilla" model checking is used to check choreography properties.

While approaches based on model checking are now relatively commonplace in services, recent prominent work [Cohn and Hull, 2009] has emphasised what appear to be severe limitations in the mainstream modelling as considered in the services literature. Specifically, it is of concern that current approaches typically focus either on the process side or on the data side, hence disregarding the interactions between these two equally relevant facets of the system. To overcome these limitations, the concepts of *artifacts* and *artifact-centric systems* (ACSs) have been put forward.

Artifacts are structures that "combine data and process in an holistic manner as the basic building block[s]" [Cohn and Hull, 2009], and ACSs are formal descriptions of (complex) workflow schemes based on artifacts.

As a result of treating both data and processes as "first-class citizens", the usual service composition questions become intrinsically much harder and can no longer been solved by current verification methodology. This is an area where solutions inspired by theoretical work in Artificial Intelligence, including knowledge representation, have, in our opinion, a fundamental role to play, particularly when combined with verification approaches.

This paper is intended to provide a first foundational stepping stone in this direction. Inspired by well-known contributions in logic and knowledge representation [Fagin *et al.*, 1995], we put forward a formalisations of artifact systems by providing them with a computationally-grounded [Wooldridge, 2000] semantics (Section 2) that we illustrate on a concrete scenario (Section 3). The semantics is used to interpret a quantified language that includes temporal and epistemic modalities to model the temporal evolution of the system's properties including the information the agents hold. In Section 4 we address the model checking problem, which, besides being interesting *per se*, provides a basis for solutions of many typical problems in this area, including orchestration and choreography. This is a difficult problem as both the quantification domain and the state-space are infinite. We make an initial, yet promising dent into this by giving abstraction results that, in some cases, permit the reduction of the problem to model checking on finite domains.

2 Formal Model

We recall the notion of database and then introduce *artifact quantified interpreted systems* (A-QIS), which we show to be well-suited to provide a semantics to ACSs.

Definition 1 (Database Schema) *A* database schema *is a set $D = \{R_1, \ldots, R_n\}$, where each R_i is a* relation schema *of the form $R_i = r_i(a_1, \ldots, a_{k_i})$, and all a_js are pairwise distinct. For each relation schema R_i we refer to r_i as the* relation *name, a_j as the j-th* relation attribute*, and k_i as R_i's (or r_i's)* arity.

Definition 2 (Database Instance) *An* instance *(or* interpretation*)* D *of a database schema D over a possibly infinite* interpretation domain V *is a set of finite relations $\mathsf{D} = \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$ such that $\mathsf{R}_i \subseteq V^{k_i}$, for $i = 1, \ldots, n$. Each R_i is called* instance *(or* interpretation*) of (relation schema) R_i.*

For a schema D, write $\mathcal{I}_D(V)$ (or simply $\mathcal{I}_D$, if V is clear from the context) for the set of all D's interpretations over V.

We capture ACSs by using ideas from Intepreted Systems semantics [Fagin *et al.*, 1995] and extensions [Belardinelli and Lomuscio, 2009].

We assume a set $Ag = \{1, \ldots, m\}$ of agents, a database schema $D = \{R_1, \ldots, R_n\}$, and an alphabet $\mathcal{A}$ containing individual constants $c_1, c_2, \ldots$, n-ary predicate letters $P_1^n, P_2^n, \ldots$ for $n \in \mathbb{N}$, as well as all relation schemes $R_1, \ldots, R_n$ in D. Further, for each agent $i \in Ag$ we introduce a set L_i of local states $l_i, l_i', \ldots$, a set ACT_i of actions $\alpha_i, \alpha_i', \ldots$, and a protocol function $P_i : L_i \longrightarrow 2^{Act_i}$. We consider local states, actions and a protocol function for the environment e as well. The set $\mathcal{S} \subseteq L_1 \times \ldots \times L_m \times L_e$ contains the global states of the systems; while $Act \subseteq Act_1 \times \cdots \times Act_m \times Act_e$ and $P = \langle P_1, \ldots, P_m, P_e \rangle$ are the set of joint actions and the joint protocol respectively. We can now define artifact quantified interpreted systems (A-QIS).

Definition 3 (A-QIS) *An* artifact quantified interpreted system *is a tuple $\mathcal{P} = \langle D, V, s_0, \tau, I \rangle$ where:*

- *for each $i \in Ag$, D_i is a view on D and $L_i = \mathcal{I}_{D_i}(V)$. Also, $L_e = \mathcal{I}_D(V)$;*
- *V is the interpretation domain of D;*
- *$s_0 \in \mathcal{S}$ is the initial global state;*
- *$\tau : \mathcal{S} \longrightarrow (Act \longrightarrow \mathcal{S})$ is the transition function, where $\tau(s)(\alpha)$ is defined only if $\alpha \in P(s)$.*
- *I is an interpretation of the alphabet $\mathcal{A}$ such that:* (i) *for every constant $c \in \mathcal{A}$, $I(c) \in V$; and* (ii) *for every predicate letter $P^n \in \mathcal{A}$, $I(P^n, s) \subseteq V^n$. In particular, for each relation schema R, $I(R, s) = \mathsf{R}$.*

For $s, s' \in \mathcal{S}$ we say that s' is a *successor* of s, or $s \longrightarrow s'$, if there exists $\alpha \in Act$ such that $s' = \tau(s)(\alpha)$. A *run* r is a sequence $r = s^0 \longrightarrow s^1 \longrightarrow \ldots$ such that $s^i \longrightarrow s^{i+1}$. For $n \in \mathbb{N}$, $r(n)$ is the n-th element in the sequence, i.e., s^n. For $s, s' \in \mathcal{S}$ we say that s is *epistemically indistinguishable* from s' for agent i, or $s \sim_i s'$, if $s_i = s_i'$ [Fagin *et al.*, 1995].

Observe that A-QISs specialise quantified interpreted systems (QIS), as defined in [Belardinelli and Lomuscio, 2009]. In fact, a QIS can be seen as an A-QIS $\mathcal{P} = \langle V, s_0, \tau, I \rangle$ with no database schema, so that the internal structure of local states is left unspecified.

2.1 The First-Order MAS Logic FO-CTLK

Given the set Ag of agents and the alphabet $\mathcal{A}$, the first-order temporal epistemic language $\mathcal{L}_m$ contains all individual constants, predicate letters and relation schema in $\mathcal{A}$, individual variables $x_1, x_2, \ldots$, the connectives $\neg$ and $\rightarrow$, the quantifier $\forall$, the branching time operators AX, AU and EU, the epistemic operator K_i for each agent $i \in Ag$, and the common knowledge operator C [Fagin *et al.*, 1995]. $\mathcal{L}_m$ contains no functional symbols, so the only terms in the language are individual variables and constants.

Definition 4 *Formulas in $\mathcal{L}_m$ are defined in BNF as follows:*

$$\phi \quad ::= \quad P^k(t_1, \ldots, t_k) \mid \neg\phi \mid \phi \rightarrow \phi \mid \forall x\phi \mid AX\phi \mid A\phi U\phi \mid E\phi U\phi \mid K_i\phi \mid C\phi$$

The formulas $AX\phi$ and $A\phi U\phi'$ (resp. $E\phi U\phi'$) are read as "*for all paths, at the next step ϕ*" and "*for all paths (resp. for some path), ϕ until ϕ'*". $K_i\phi$ means "*agent i knows ϕ*"; while $C\phi$ is read as "*ϕ is common knowledge*". The logical symbols $\wedge$, $\vee$, $\leftrightarrow$ and $\exists$, as well as the operators AG, AF, EG, EF, and EX are defined as standard. Finally, $E\phi$ is defined as $\bigwedge_{i \in Ag} K_i\phi$, and for $n \in \mathbb{N}$, $E^0\phi = \phi$ and $E^{n+1}\phi = EE^n\phi$.

By $\phi[\vec{y}]$ we mean that $\vec{y} = y_1, \ldots, y_n$ are all ϕ's free variables; and $\phi[\vec{y}/\vec{t}]$ is the formula obtained by substituting simultaneously some, possibly all, free occurrences of $\vec{y}$ in ϕ with $\vec{t} = t_1, \ldots, t_n$ while renaming bound variables. As standard, a sentence is a formula with no free variables.

In what follows we consider two fragments of $\mathcal{L}_m$.

Definition 5 *Formulas in the $\forall$ACTLK-fragment of $\mathcal{L}_m$ are defined in BNF as follows, where the logical symbols $\vee$, $\wedge$, $A\overline{U}$, and $\exists$ are taken as primitives:*

$$\phi \quad ::= \quad P^k(\vec{t}) \mid \neg P^k(\vec{t}) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \forall x\phi \mid AX\phi \mid A\phi U\phi \mid A\phi\overline{U}\phi \mid K_i\phi \mid C\phi$$

where $A\phi\overline{U}\phi'$ is read as "for all paths, ϕ release ϕ'"*.*

The $ACTLK$-fragment extends the $\forall ACTLK$-fragment with the following clause:

- *if ϕ is a formula, then $\exists x\phi$ is also a formula.*

We now define the semantics of $\mathcal{L}_m$ formulas in terms of A-QISs. Given an assignment σ from the set of variables in $\mathcal{L}_m$ to the individuals in V, the interpretation $I^\sigma(t)$ of an individual term t is defined as $\sigma(t)$ if t is a variable, or $I(t)$ if t is a constant. Also, σ_a^x is the assignment that maps x to a and coincides with σ on all other variables.

Definition 6 *The satisfaction relation $\models$ for $\phi \in \mathcal{L}_m$, $s \in \mathcal{S}$, and an assignment σ is inductively defined as follows:*

$(\mathcal{P}^\sigma, s) \models P^k(\vec{t})$ *iff* $\langle I^\sigma(t_1), \ldots, I^\sigma(t_k) \rangle \in I(P^k, s)$
$(\mathcal{P}^\sigma, s) \models \neg\phi$ *iff* $(\mathcal{P}^\sigma, s) \not\models \phi$
$(\mathcal{P}^\sigma, s) \models \phi \rightarrow \phi'$ *iff* $(\mathcal{P}^\sigma, s) \not\models \phi$ *or* $(\mathcal{P}^\sigma, s) \models \phi$
$(\mathcal{P}^\sigma, s) \models \forall x\phi$ *iff for all* $a \in V$, $(\mathcal{P}^{\sigma_a^x}, s) \models \phi$
$(\mathcal{P}^\sigma, s) \models AX\phi$ *iff for all runs r, if* $r(n) = s$ *then* $(\mathcal{P}^\sigma, r(n+1)) \models \phi$
$(\mathcal{P}^\sigma, s) \models A\phi U\phi'$ *iff for all runs r, if* $r(n) = s$ *then there is* $k \geq n$, $(\mathcal{P}^\sigma, r(k)) \models \phi'$ *and* $n \leq k' < k$ *implies* $(\mathcal{P}^\sigma, r(k')) \models \phi$
$(\mathcal{P}^\sigma, s) \models E\phi U\phi'$ *iff for some run r,* $r(n) = s$ *and there is* $k \geq n$, $(\mathcal{P}^\sigma, r(k)) \models \phi'$ *and* $n \leq k' < k$ *implies* $(\mathcal{P}^\sigma, r(k')) \models \phi$
$(\mathcal{P}^\sigma, s) \models K_i\phi$ *iff for all* s, s', *if* $s \sim_i s'$ *then* $(\mathcal{P}^\sigma, s') \models \phi$
$(\mathcal{P}^\sigma, s) \models C\phi$ *iff for all* $k \in \mathbb{N}$, $(\mathcal{P}^\sigma, s) \models E^k\phi$

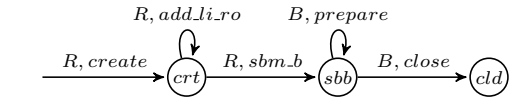
A formula $\phi \in \mathcal{L}_m$ is *true* in a state s, or $(\mathcal{P}, s) \models \phi$, if for all assignment σ, $(\mathcal{P}^\sigma, s) \models \phi$; it is *true* in an A-QIS $\mathcal{P}$, or $\mathcal{P} \models \phi$, if $(\mathcal{P}, s_0) \models \phi$.

3 The Equipment Purchasing Scenario

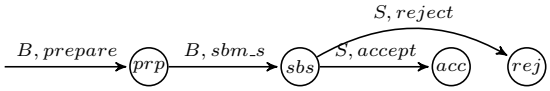
We now introduce a scenario, inspired by a real Business Process use-case developed at IBM, which shows the artifact-centric approach at work. Although we do not formally define a general translation procedure from artifact-centric business processes to A-QIS, this formalisation exemplifies how this can be done and, therefore, demonstrates that A-QISs can effectively be employed to reason about real ACSs.

3.1 The Scenario

The employees of a company who need to purchase some equipment must follow a procedure, which involves the following agents: a *requester*, who needs to purchase the equipments; a *buyer*, who is in charge of buying the requested



(a) Request order lifecyle



(b) Procurement order lifecyle

Figure 1: Lifecycles of request and procurement orders

items; and some *suppliers*, who supply the equipment. The process begins when the requester fills out a *requisition order* with some line items referring to desired products (e.g., a laptop). The requester then submits the order to the buyer, who finds an appropriate supplier for each line item, and prepares a *procurement order* containing the relevant line items. Procurement orders are then submitted to suppliers, who can either reject or fulfil the received orders. In the former case, the buyer is notified; in the latter, the items are shipped.

Orders are captured by *artifacts*, i.e., structures containing data fields, and associated with sets of actions enabling data manipulation. Requisition orders contain fields: id, the requisition order identifier; itm, the line item(s) occurring in the order; p_ord, the procurement order(s) associated with the requisition order; and $status$ (see below). Procurement orders contain: id, the procurement order identifier; s_id, the identifier of the supplier selected by the buyer; itm, the line item(s) occurring in the order; and $status$ (see below).

The evolution of an artifact $status$ field, in response to agent actions, is referred to as the artifact's *lifecyle*, which accounts for the stage of the process that the artifact is in. Figures 1(a) and 1(b) depict the lifecycles for request and procurement orders in the form of transition systems, where nodes are labelled by the status they represent, and transitions by pairs "ag, op", associating each action (op) with the agent (ag) that can execute it, R standing for requester, B for buyer, and S for supplier. Each request order has 3 possible statuses, namely crt (created), sbb (submitted to buyer), and cld (closed), and 5 transitions: $R, create$ (the requester creates a request order); R, add_li_ro (the requester adds a line item to the created request order); R, sbm_b (the requester submits the order to the buyer); $B, prepare$ (the requester prepares a procurement order for this request order); $B, close$ (the requester closes the request order, when all items are shipped). The lifecycle of procurement orders is similar, though not described here for brevity.

3.2 Formalisation

For simplicity we assume each requisition order contains at most one item; however, A-QISs are sufficiently expressive to overcome this limitation.

Agents. The set of agents is $Agt = \{1, 2, \ldots, n_S, e\}$, where: 1 represents the requester; 2 the buyer; $3, \ldots, n_S$ stand for the suppliers; and e represents the environment.

Interpretation Domain. We take $V = ID_A \cup ID_S \cup LI$ as the interpretation domain, where: ID_A is an infinite set of *artifact identifiers*, $ID_S = \{3, \ldots, n_S\}$ is the finite set of *supplier identifiers* (see below), and LI is the finite set of all line items, i.e. $LI = \{monitor, printer, phone\}$. All these sets are assumed disjoint. Moreover, we take the symbol $- \in V$ to represent a null value.

Environment Database Schema. We define the A-QIS environment database schema $D_e = \{RO, PO\}$, where the relations RO and PO are intended to record the information about all request and procurement orders. Their schemes are as follows: $RO(id, itm, p\_ord, status)$, where: id is the requisition order identifier, itm is the (single) line item occurring in the order, p_ord is the (single) procurement order associated with the requisition order, and $status$ is the current status of the order; $PO(id, s\_id, itm, status)$, where: id is the procurement order identifier, s_id is the identifier of the supplier selected by the buyer, itm is the (single) line item in the order, and $status$ is the current order status.

Local States. To model the requester's (i.e., agent 1) local-state space we take the set $\mathcal{I}_{D_1}(V)$ of interpretations of the database $D_1 = \{RO_1\}$, where RO_1's schema matches RO's. Similarly, buyer's (i.e., agent 2) local states are interpretations of the database $D_2 = \{RO_2, PO_2\}$ over V ($\mathcal{I}_{D_2}(V)$) where RO_2's and PO_2's schemes match RO's and PO's, respectively. As for agents $3, \ldots, n_S$, the suppliers, local states are interpretations of databases $D_i = \{PO_i\}$ over V, i.e., $\mathcal{I}_{D_i}(V)$, for $i = 3, \ldots, n_S$, where PO_i's schema is the same as PO's Finally, as set of environment local states we simply take $\mathcal{I}_{D_e}(V)$.

Actions. For conciseness, without loss of generality, we use parametric actions. A parametric action of the form $a(p_1, \ldots, p_q)$, where a is the action name and $p_1, \ldots, p_q$ are its parameters, represents a family of actions containing one *ground* action per distinct parameter assignment of values from V. As a convention, for an action parameter p, we denote the generic value assigned to p as p. Agents' action sets are defined as follows. For agent 1, we take $Act_1 = \{create, add\_li\_ro(id, li), sbm\_b(id)\}$, where: $create$ is meant to create a new request order (with unique identifier); $add\_li\_ro(id, li)$ is meant to add line item li to the request order identified by id; and $sbm\_b(id)$ is meant to submit the request order identified by id to the buyer. The action sets for the other agents are similar, but omitted here. As for the environment, we simply have $Act_e = \emptyset$.

Protocol Functions. We only report the buyer's protocol function, the other agents' are similar, while, Act_e being empty, no protocol is defined for the environment. The buyer's protocol $P_2 : \mathcal{I}_{D_2}(V) \longrightarrow 2^{Act_2}$ is as follows ($\mathsf{D}_2 = \{\mathsf{RO}_2, \mathsf{PO}_2\}$). $prepare(\mathsf{id}, \mathsf{s\_id}) \in P_2(\mathsf{D}_2)$ if there exists a tuple $\langle \mathsf{id}, \mathsf{itm}, \mathsf{p\_ord}, \mathsf{sbb} \rangle \in \mathsf{RO}_2$, and either $\mathsf{p\_ord} = -$ or there exists a tuple $\langle \mathsf{p\_ord}, \mathsf{s\_id}, \mathsf{itm}, \mathsf{s} \rangle \in \mathsf{PO}_2$ s.t. $\mathsf{s} \in \{\mathsf{prp}, \mathsf{rej}\}$. This formalises that the buyer can prepare a new procurement order for an existing request order only if the request order is in status sbb, and either no corresponding procurement order has been prepared yet, or, if it has, it is either in status prp (prepared) or rej (rejected). $close(\mathsf{id}) \in P_2(\mathsf{D}_2)$ if there exists two tuples: $\langle \mathsf{id}, \mathsf{itm}, \mathsf{p\_ord}, \mathsf{sbb} \rangle \in \mathsf{RO}_2$ with $\mathsf{p\_ord} \neq -$, and $\langle \mathsf{p\_ord}, \mathsf{itm}, \mathsf{s\_id}, \mathsf{acc} \rangle \in \mathsf{PO}_2$, that is, the buyer can close only existing request orders whose associated procurement order is accepted; and $sbm\_s(\mathsf{id}) \in P_2(\mathsf{D}_2)$

if there exists a tuple $\langle \mathsf{id}, \mathsf{s\_id}, \mathsf{itm}, \mathsf{prp} \rangle \in \mathsf{PO}_2$, i.e., the buyer can submit (to a supplier) only prepared procurement orders.

Interpretation Function. The interpretation function I is simply defined as the identity on the environment's local state, that is, for $s = \langle \mathsf{D}_1, \mathsf{D}_2, \mathsf{D}_3, \ldots, \mathsf{D}_{n_S}, \mathsf{D}_e \rangle \in S$, $I(s) = \mathsf{D_e}$.

Initial State. As for the (global) initial state, we assume all relations are initially empty.

Global Transition Function. First, we define some dependencies between the agents' local states and the environment's. For a global state $s = \langle \mathsf{D}_1, \mathsf{D}_2, \mathsf{D}_3, \ldots, \mathsf{D}_{n_S}, \mathsf{D}_e \rangle$, with $\mathsf{D}_1 = \{\mathsf{RO}_1\}$, $\mathsf{D}_i = \{\mathsf{PO}_i\}$, for $i = 3, \ldots, n_S$, and $\mathsf{D}_e = \{\mathsf{RO}, \mathsf{PO}\}$, we have: $\mathsf{RO}_1 = \mathsf{RO}$, $\mathsf{D}_2 = \mathsf{D}_e$, and for each $i = 3, \ldots, n_S$, $\langle \mathsf{id}, \mathsf{s\_id}, \mathsf{itm}, \mathsf{status} \rangle \in \mathsf{PO}_i$ iff $\mathsf{s\_id} = i$ and $\langle \mathsf{id}, \mathsf{s\_id}, \mathsf{itm}, \mathsf{status} \rangle \in \mathsf{PO}$. For simplicity, given a current global state s and a joint action a, we define the successor global state $s' = \tau(a)(s)$ by defining its environment component $\mathsf{D}'_e = \{\mathsf{RO}', \mathsf{PO}'\}$, as all other components are derivable through the above dependencies. For brevity, we describe only one action.

For $a = \langle create, -, -, \ldots, - \rangle$, D'_e in s' is such that $\mathsf{PO}'_e = \mathsf{PO}_e$, and $\mathsf{RO}'_e = \mathsf{RO}_e \cup \{\langle \mathsf{id}', -, -, crt \rangle\}$, with $\mathsf{id}' \in ID_A$ such that there is no other tuple $\langle \mathsf{id}, \mathsf{itm}, \mathsf{p\_ord}, \mathsf{status} \rangle \in \mathsf{RO}$ such that $\mathsf{id} = \mathsf{id}'$. Informally, when the requester creates a new order, a new request order artifact is created, with a unique identifier.

Once the system has been modelled, it is possible to verify particular specifications on it. For instance, the system described above satisfies formula $\varphi_1 = AG(\forall id_r, itm, p \quad RO_e(id_r, itm, p, \mathsf{cld}) \rightarrow \exists s \; K_2 PO_e(p, s, itm, \mathsf{acc}))$, which states that a request order can be in state *closed* only if the buyer knows that the corresponding procurement order has been actually accepted by some supplier (this corresponds to the precondition of action $close$ according to protocol P_2). This property intuitively corresponds to a specification that should indeed be satisfied in the scenario considered. Other epistemic specifications of interest can be similarly formalised.

4 Model checking and Abstraction

We would like to be able to verify automatically any formula φ in $\mathcal{L}$ on a given A-QIS, i.e., to give an effective methodology for answering the model checking query $\mathcal{P} \models \varphi$. The (considerable) difficulty resides in the fact that $\mathcal{P}$ is, in general, an infinite structure. To make inroads into this problem we give an abstraction technique for A-QIS by extending the results presented in [Cohen *et al.*, 2009] to the case of infinite models. We present the results obtained in their fullest generality by giving them on structures built on arbitrary sets, rather than specific database views (Def. 3). This corresponds to the general class of quantified interpreted systems (QIS) as defined in [Belardinelli and Lomuscio, 2009]. Given A-QIS are a subclass of QIS, all the results here proved also hold for A-QIS.

4.1 Simulation

The standard notion of simulation for reactive systems states that a system simulates another if every behaviour of the latter is a behaviour of the former [Clarke *et al.*, 1994]. Since ACTL operators quantify over all behaviours (runs), any ACTL property that holds in the simulating system holds also in the simulated system. To extend this preservation property to ∀ACTLK, we require that any epistemic possibility in the simulated system is matched by an epistemic possibility in the simulating system. Similar conditions apply to individuals.

Definition 7 (Sublanguage) *Let $\mathcal{L}$ and $\mathcal{L}'$ be first-order temporal epistemic languages as in Def. 4, with alphabets $\mathcal{A}$ and $\mathcal{A}'$ respectively. $\mathcal{L}'$ is a sub-language of $\mathcal{L}$, or $\mathcal{L}' \subseteq \mathcal{L}$, if $\mathcal{A}' \subseteq \mathcal{A}$.*

We now define the notion of simulation for QIS.

Definition 8 (Simulation) *Let $\mathcal{P} = \langle V, s_0, \tau, I \rangle$ be a QIS on the set Ag of agents and the language $\mathcal{L}$, and let $\mathcal{P}' = \langle V', s'_0, \tau', I' \rangle$ be a QIS on the same set Ag of agents and a sub-language $\mathcal{L}' \subseteq \mathcal{L}$. A simulation between $\mathcal{P}$ and $\mathcal{P}'$ is a pair of relations $\simeq \subseteq \mathcal{S} \times \mathcal{S}'$ and $\approx \subseteq V \times V'$ such that:*

(a) $s_0 \simeq s'_0$;

(b) if $a \in V$ then there exists $a' \in V'$ such that $a \approx a'$;

and if $s \simeq s'$ then:

(c) if $s \longrightarrow u$ then $s' \longrightarrow' u'$ for some u' such that $u \simeq u'$;

(d) if $s \sim_i u$ then $s' \sim'_i u'$ for some u' such that $u \simeq u'$;

(e) for all $P^n \in \mathcal{L}'$, if $\vec{a} \approx \vec{a}'$ then $\vec{a} \in I(P^n, s)$ iff $\vec{a}' \in I'(P^n, s')$;

(f) for all $c \in \mathcal{L}'$, $I(c) \approx I'(c)$;

where $\longrightarrow$ and $\longrightarrow'$ are the transition relations in $\mathcal{P}$ and $\mathcal{P}'$ respectively. If there is a simulation pair between $\mathcal{P}$ and $\mathcal{P}'$, we say that $\mathcal{P}'$ simulates $\mathcal{P}$, or $\mathcal{P} \preceq \mathcal{P}'$.

According to (a) the initial state in $\mathcal{P}$ has to be matched by the initial state in $\mathcal{P}'$. Similarly for (b) and individuals. According to (c) and (d) every temporal and epistemic transition in $\mathcal{P}$ has to be matched by a transition in $\mathcal{P}'$. According to (e) and (f) related states must agree on the sub-language $\mathcal{L}'$.

Any ∀ACTLK property is preserved from the simulating QIS $\mathcal{P}'$ to the QIS $\mathcal{P}$ being simulated:

Lemma 1 *Assume that $\mathcal{P}'$ simulates $\mathcal{P}$. For any ∀ACTLK-formula $\phi \in \mathcal{L}'$, if $\mathcal{P}' \models \phi$ then $\mathcal{P} \models \phi$.*

Lemma 1 follows directly from the following remark:

$$\text{if } (\mathcal{P}'^{\sigma'}, s') \models \phi, s \simeq s' \text{ and } \sigma(x) \approx \sigma'(x) \text{ then } (\mathcal{P}^{\sigma}, s) \models \phi \tag{1}$$

The proof, omitted here, is by induction on the length of ϕ.

Differently from the propositional level, we can define a strengthening of the notion of simulation, so that also existential formulas are also preserved.

Definition 9 (Simulation$^+$) *A simulation$^+$ between $\mathcal{P}$ and $\mathcal{P}'$ is a pair of relations $\simeq \subseteq \mathcal{S} \times \mathcal{S}'$ and $\approx \subseteq V \times V'$ such that:*

(a) $\simeq$ and $\approx$ are a simulation pair between $\mathcal{P}$ and $\mathcal{P}'$;

(b) if $a' \in V'$ then there exists $a \in V$ such that $a \approx a'$.

If there is a simulation$^+$ pair between $\mathcal{P}$ and $\mathcal{P}'$, we say that $\mathcal{P}'$ simulates$^+$ $\mathcal{P}$, or $\mathcal{P} \preceq^+ \mathcal{P}'$.

We can now prove the following strengthening of Lemma 1.

Lemma 2 *Assume that $\mathcal{P}'$ simulates$^+$ $\mathcal{P}$. For any ACTLK-formula $\phi \in \mathcal{L}'$, if $\mathcal{P}' \models \phi$ then $\mathcal{P} \models \phi$.*

The result follows from (1), where the inductive case for the existential quantifier makes use of clause (b) in Def. 9. We omit the complete proof for reasons of space.

In the rest of the paper we focus on simulation$^+$ and the ACTLK-fragment.

4.2 Existential Abstraction of QIS

In systems with large state spaces, it is infeasible to verify design requirements by considering all reachable states, even if represented symbolically [Burch *et al.*, 1992]. In existential abstraction [Clarke *et al.*, 1994], one reduces a large, possibly infinite reactive system - referred to as the *concrete* system - into a possibly smaller reactive system - referred to as the *abstract* system - by partitioning the system states into equivalence classes. Each equivalence class, called an *abstract state*, forms a state in the abstract system.

Here, we extend existential abstraction to quantified interpreted systems by abstracting each agent i and the quantification domain V separately. Formally, the abstract QIS is defined as a quotient construction as follows. Assume a quantified interpreted system $\mathcal{P}$ over the set Ag of agents and the language $\mathcal{L}$. For each $i \in Ag$ assume the equivalence relations $\equiv_i \subseteq L_i \times L_i$ and $\equiv_i \subseteq ACT_i \times ACT_i$. Further, assume an equivalence relation $\equiv \subseteq V \times V$. For $l \in L_i$, write $[l]$ for the equivalence class of l w.r.t. $\equiv_i$, and $[\alpha]$ for the equivalence class of $\alpha \in ACT_i$ w.r.t. $\equiv_i$. Similarly, $[a]$ is the equivalence class of $a \in V$ w.r.t. $\equiv$. Write $[s]$ for $\langle [s_1], \ldots, [s_n] \rangle$ and write $[\vec{\alpha}]$ for $\langle [\alpha_1], \ldots, [\alpha_n] \rangle$. Finally, let $\mathcal{L}' \subseteq \mathcal{L}$ be a sub-language of $\mathcal{L}$ that does not distinguish between equivalent local states and individuals, i.e.,

(*) for all $P^n \in \mathcal{L}'$, if $s \equiv s'$ and $\vec{a} \equiv \vec{a}'$ then $\vec{a} \in I(P^n, s)$ iff $\vec{a}' \in I(P^n, s')$.

Definition 10 (Quotient QIS) *The quotient of $\mathcal{P}$ is the QIS $\mathcal{P}'$ on the set Ag of agents and the sub-language $\mathcal{L}' \subseteq \mathcal{L}$ such that:*

1. $V' = \{[a] \mid a \in V\}$;
2. $L'_i = \{[l] \mid l \in L_i\}$;
3. $ACT'_i = \{[\alpha] \mid \alpha \in ACT_i\}$;
4. $P'_i = \{\langle [l], [\alpha] \rangle \mid \langle l, \alpha \rangle \in P_i\}$;
5. $\tau' = \{\langle [s], [\vec{\alpha}], [s'] \rangle \mid \langle s, \vec{\alpha}, s' \rangle \in \tau\}$;
6. $s'_0 = [s_0]$;
7. *for all* $P^n \in \mathcal{L}'$, $[\vec{a}] \in I'(P^n, [s])$ *iff* $\vec{a} \in I(P^n, s)$;
8. *for all* $c \in \mathcal{L}'$, $I'(c) = [I(c)]$.

Note that the interpretation I is well defined by condition (*) on the sub-language $\mathcal{L}'$. Observe that Def. 10 does not specify how the equivalence relations are chosen. This issue is addressed in Section 4.3 below. The important property of quotient systems, however, is that specifications are preserved from abstract systems to concrete ones. This is because the abstract system simulates$^+$ the original system.

Lemma 3 *If $\mathcal{P}'$ is a quotient of $\mathcal{P}$, then $\mathcal{P}'$ simulates$^+$ $\mathcal{P}$.*

Proof sketch. We show that the relations $\simeq = \{\langle s, [s] \rangle \mid s \in V\}$ and $\approx = \{\langle a, [a] \rangle \mid a \in V\}$ are a simulation$^+$ pair for $\mathcal{P}$ and $\mathcal{P}'$. Simulation requirements (a) and (b) follow from 6 and 1 in Def. 10 respectively. Requirement (c) follows from 4 and 5; while requirement (d) follows by the definition of equivalence classes. Simulation requirements (e) and (f) follow from 7 and 8 respectively. Finally, the simulation$^+$ requirements (b) in Def. 9 trivially hold. □

Since the abstract system simulates$^+$ the concrete system, design requirements expressed in $ACTLK$ are preserved.

Theorem 4 (Preservation) *Let $\mathcal{P}'$ be a quotient of the QIS $\mathcal{P}$. For any ACTLK-formula ϕ in $\mathcal{L}' \subseteq \mathcal{L}$, if $\mathcal{P}' \models \phi$ then $\mathcal{P} \models \phi$.*

Proof. From Lemma 2 and Lemma 3. □

When applying the theorem, the challenge is to choose suitable equivalence relations $\equiv_i$ on local states and actions. We provide a partial answer in the following section.

4.3 Constructive Abstraction

In this section we introduce a methodology for defining the equivalence relations defined in the previous section. In what follows we fix a sub-language $\mathcal{L}'$ of the first-order temporal epistemic language $\mathcal{L}_m$, and a QIS $\mathcal{P}$. We first define equivalences on the set V^n of n-tuples of individuals.

Definition 11 (Equivalence on tuples) *Let $s \in \mathcal{S}$ and let $\vec{a}$, $\vec{b}$ be n-tuples in V^n. We say that $\vec{a}$ and $\vec{b}$ are equivalent in s, or $\vec{a} \sim_s \vec{b}$, if for all $P^n \in \mathcal{L}'$, $\vec{a} \in I(P^n, s)$ iff $\vec{b} \in I(P^n, s)$.*

We can easily check that the relation $\sim_s$ is indeed an equivalence relation for every $s \in \mathcal{S}$.

Definition 12 (Equivalence on individuals) *Let $s \in \mathcal{S}$ and let a, b be individuals in V. We say that a and b are equivalent in s, or $a \equiv_s b$, if for all $\vec{a}$, $\vec{a}' \in V^n$, if $\vec{a}'$ is obtained from $\vec{a}$ by uniformly substituting a with b, then $\vec{a} \sim_s \vec{a}'$.*

From the fact that $\sim_s$ is an equivalence relation we can derive that also $\equiv_s$ is an equivalence relation for $s \in \mathcal{S}$.

We now define the equivalence relation on states.

Definition 13 (Equivalence on states) *Two states s, $s' \in \mathcal{S}$ are equivalent , or $s \equiv s'$, if for all $P^n \in \mathcal{L}'$, for all $\vec{a} \in V^n$, $\vec{a} \in I(P^n, s)$ iff $\vec{a} \in I(P^n, s')$.*

We can now introduce the abstract model obtained from the QIS $\mathcal{P}$.

Definition 14 (Abstract model) *Given the QIS $\mathcal{P} = \langle V, s_0, \tau, I \rangle$ we define an abstract model $\mathcal{M}' = \langle V', [s_0], \tau', I' \rangle$ such that:*

- $\mathcal{S}' = \{[s] \mid s \in \mathcal{S}\}$;
- *for each* $[s] \in \mathcal{S}'$, $V'([s]) = \{[a]_s \mid a \in V\}$;
- *for any* $[s], [s'] \in \mathcal{S}$, $[s] \longrightarrow [s']$ *if* $s \longrightarrow s'$;
- *for* $P^n \in \mathcal{L}'$, $[\vec{a}]_s \in I'(P^n, [s])$ *iff* $\vec{a} \in I(P^n, s)$.

By definition of the individuals in each $V'([s])$ we can prove that the interpretation I' is well-defined and independent from the particular choice of representatives for the equivalence classes in $V'([s])$.

Hereafter we introduce the satisfaction relation for the abstract model.

Definition 15 *The satisfaction relation $\models$ for $\phi \in \mathcal{L}'$, $[s] \in \mathcal{M}'$, and an assignment σ is defined as in Def. 6, but for the following clauses:*

$(\mathcal{M}'^{\sigma}, [s]) \models P^k(\vec{t})$ *iff* $\langle [I'^{\sigma}(t_1)]_s, \ldots, [I'^{\sigma}(t_k)]_s \rangle \in I'(P^k, [s])$
$(\mathcal{M}'^{\sigma}, [s]) \models \forall x \psi$ *iff for all* $a \in V([s])$, $(\mathcal{M}'^{\sigma^x_a}, [s]) \models \psi$

Now, we can prove the following result on $\mathcal{M}'$.

Lemma 5 *The abstract model $\mathcal{M}'$ simulates$^+$ $\mathcal{P}$*

Proof sketch. The proof consists in showing that the relations $\simeq = \{\langle s, [s] \rangle \mid s \in \mathcal{S}\}$ and $\approx = \{\langle a, [a]_s \rangle \mid a \in V, s \in \mathcal{S}\}$ are a simulation$^+$ pair for $\mathcal{P}$ and $\mathcal{M}'$, and it is similar to Lemma 3. □

The next result is immediate from Lemmas 2 and 5.

Lemma 6 *For every ACTLK-formula ϕ in $\mathcal{L}'$, if $\mathcal{M}' \models \phi$ then $\mathcal{P} \models \phi$.*

Thus, we have obtained an effective way of constructing the quotient system $\mathcal{M}'$ starting from $\mathcal{P}$ and $\mathcal{L}'$. More in detail, given a ∀ACTLK-formula ϕ to be model checked on $\mathcal{P}$, $\mathcal{L}'$ can be thought of as the sub-language of $\mathcal{L}$ consisting of all predicate letters and constants in ϕ.

Finally, we observe that if a QIS $\mathcal{P}$ has infinitely many states and individuals, then also the abstract model $\mathcal{M}'$ will in general be infinite both in states and individuals. However, the construction above does in some cases generate finite approximations.

Lemma 7 *Given a QIS $\mathcal{P}$, its abstract model $\mathcal{M}'$ has the following cardinality:*

$\mathcal{P}$		$\mathcal{M}'$	
$\mathcal{S}$	V	$\mathcal{S}'$	V'
infinite	*infinite*	*infinite*	*infinite*
finite	*infinite*	*finite*	*infinite*
infinite	*finite*	*finite*	*finite*
finite	*finite*	*finite*	*finite*

The proof is immediate by definition of $\mathcal{M}'$ from $\mathcal{P}$.

We have therefore identified a non-trivial case (infinite $\mathcal{S}$ and finite V) in which the technique presented above generates a feasible model checking problem. Additionally, observe that in the case of finite $\mathcal{S}$ and infinite V if all state interpretations are finite (e.g., I maps states into database instances) we also obtain, as result of the abstraction process, a finite V' in M'.

5 Conclusions

In this paper we have put forward a computationally-grounded semantics for artifact systems and illustrated its use in the context of a temporal epistemic specification language. This is intended to provide a basis for model checking these systems in combination with suitable abstraction techniques. We have provided a first set of results in this context and shown that, at least in certain cases of interest, finite abstractions of artifact-systems can be obtained. A limitation of the results presented is that not in all cases finite abstractions are generated. In verification this is known to be a very severe challenge and is therefore to be expected to appear here as well. To surmount these difficulties, our future work includes the development of techniques such as *data independence* [Wolper, 1986] in the context of A-QIS, and the investigation on abstraction schemes similar to [Deutsch *et al.*, 2009] by relaxing the completeness requirement.

In service-oriented computing, to the best of our knowledge, very few attempts exist to reduce data-aware infinite state systems to finite ones. Noteworthy examples are [Deutsch *et al.*, 2009], [Berardi *et al.*, 2005], and [Cangialosi *et al.*, 2010], where different restrictions on input models and less expressive specifications are used. Furthermore, these works focus on finding decidable fragments from the synthesis and verification point of view, whereas the approach here presented offers a first foundational basis for data abstraction in ACSs for practical model checking.

Acknowledgements. The research leading to these results has received funding from the EC FP7 under grant agreements n. 235329 and n. 257593, and from the EPRSC grant EP/I00520X.

References

[Alonso *et al.*, 2004] G. Alonso, F. Casati, H. A. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, 2004.

[Belardinelli and Lomuscio, 2009] F. Belardinelli and A. Lomuscio. Quantified Epistemic Logics for Reasoning About Knowledge in Multi-Agent Systems. *Artificial Intelligence*, 173(9-10):982–1013, 2009.

[Berardi *et al.*, 2005] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic Composition of Transition-based Semantic Web Services with Messaging. In *VLDB*, pages 613–624, 2005.

[Burch *et al.*, 1992] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, 98(2):142–170, 1992.

[Cangialosi *et al.*, 2010] P. Cangialosi, G. De Giacomo, R. De Masellis, and R. Rosati. Conjunctive Artifact-Centric Services. In *ICSOC*, pages 318–333, 2010.

[Clarke *et al.*, 1994] E. M. Clarke, O. Grumberg, and D. Long. Model Checking and Abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[Cohen *et al.*, 2009] M. Cohen, M. Dam, A. Lomuscio, and F. Russo. Abstraction in Model Checking Multi-Agent Systems. In *AAMAS (2)*, pages 945–952, 2009.

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

[Deutsch *et al.*, 2009] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic Verification of Data-centric Business Processes. In *ICDT*, pages 252–267, 2009.

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

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

[Wooldridge, 2000] M. Wooldridge. Computationally Grounded Theories of Agency. In *Proc. of ICMAS*, pages 13–22. IEEE Press, 2000.