An Abstraction Technique for the Verification of Artifact-Centric Systems

Francesco Belardinelli
Laboratoire IBISC, Université d’Evry

Joint work with Alessio Lomuscio
Imperial College London, UK

and Fabio Patrizi
Sapienza Università di Roma, Italy

within the EU funded project ACSI (Artifact-Centric Service Interoperation)

JAIF – 13 June 2013
Model Checking in one slide

Model checking: technique(s) to **automatically** verify that a system design $S$ satisfies a property $P$ **before** deployment.

More formally, given
- a model $\mathcal{M}_S$ of a system $S$
- a formula $\phi_P$ representing a property $P$

we check that

$$\mathcal{M}_S \models \phi_P$$
Turing Award 2007

For their roles in developing model checking into a highly effective verification technology, widely adopted in the hardware and software industries.

(a) E. Clarke (CMU, USA)
(b) A. Emerson (U. Texas, USA)
(c) J. Sifakis (IMAG, F)
Motivation: Artifact Systems are *data-aware* systems

Main task: *formal verification of infinite-state* AS

- model checking is appropriate for control-intensive applications...
- ...but less suited for data-intensive applications (data typically ranges over infinite domains) [1].

Key contribution: verification of *bounded* and *uniform* AS is decidable
Artifact Systems

Outline

• Recent paradigm for Service-Oriented Computing [2].
• **Motto:** let’s give *data* and *processes* the same relevance!
• **Artifact:** data model + lifecycle
  ▶ (nested) records equipped with actions
  ▶ actions may affect several artifacts
  ▶ evolution stemming from the interaction with other artifacts/external actors
• **Artifact System:** set of interacting artifacts, representing services, manipulated by agents.
Artifcats Systems
Order-to-Cash Scenario

Customer

<table>
<thead>
<tr>
<th>Purchase Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desk</td>
</tr>
<tr>
<td>Chair</td>
</tr>
</tbody>
</table>

Manufacturer

<table>
<thead>
<tr>
<th>Material Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desk Legs</td>
</tr>
<tr>
<td>Chair Legs</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Supplier

<table>
<thead>
<tr>
<th>Material Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hammer Nails</td>
</tr>
<tr>
<td>Glue</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Research questions

1. Which syntax and semantics should we use to specify AS?
2. Is verification of AS decidable?
3. If not, can we identify relevant fragments that are reasonably well-behaved?
4. How can we implement this?
Challenges

Multi-agent systems, but . . .

- . . . states have a relational structure,
- data are potentially infinite,
- state space is infinite in general.

⇒ The model checking problem cannot be tackled by standard techniques.
**Artifact Systems**

**Results**

1. *Artifact-centric multi-agent systems (AC-MAS)*: formal model for AS.
   **Intuition**: databases that evolve in time and are manipulated by agents.

2. FO-CTLK as a specification language:

   \[
   AG \forall id, pc (\exists \vec{\bar{x}} \ MO(id, pc, \vec{\bar{x}}) \rightarrow K_M \ \exists \vec{\bar{y}} \ PO(id, pc, \vec{\bar{y}}))
   \]

   the manufacturer M knows that each MO has to match a corresponding PO.

3. Abstraction techniques and finite interpretation to tackle model checking.
   **Main result**: under specific conditions MC can be reduced to the finite case.
The data model of Artifact Systems is given as a database.

- a **database schema** is a finite set \( \mathcal{D} = \{ P_1/a_1, \ldots, P_n/a_n \} \) of predicate symbols \( P_i \) with arity \( a_i \in \mathbb{N} \).
- an **instance** on a domain \( U \) is a mapping \( D \) associating each predicate symbol \( P_i \) with a finite \( a_i \)-ary relation on \( U \).
- the **active domain** \( \text{adom}(D) \) is the set of all \( u \in U \) appearing in \( D \)
- **Composition**: \( D \oplus D' \) is the \( (\mathcal{D} \cup \mathcal{D}') \)-interpretation s.t.
  (i) \( D \oplus D'(P_i) = D(P_i) \), and
  (ii) \( D \oplus D'(P'_i) = D'(P_i) \).
Agents have partial access (views) to the artifact system.

- an agent is a tuple \( i = \langle D_i, Act_i, Pr_i \rangle \) where
  - \( D_i \) is the local database schema
  - \( Act_i \) is the set of local actions \( \alpha(\vec{x}) \) with parameters \( \vec{x} \)
  - \( Pr_i : D_i(U) \mapsto 2^{Act_i(U)} \) is the local protocol function

- the setting is reminiscent of the interpreted systems semantics for MAS [3],...
- ...but here the local state of each agent is relational.

Intuitively, agents manipulate artifacts and have (partial) access to the information contained in the global db schema \( D \).
Example 1: the Order-to-Cash Scenario

- Agents: Customer, Manufacturer, Supplier.
- Local db schema $\mathcal{D}_C$
  - $Products(prod\_code, budget)$
  - $PO(id, prod\_code, offer, status)$
- Local db schema $\mathcal{D}_M$
  - $PO(id, prod\_code, offer, status)$
  - $MO(id, prod\_code, price, status)$
- Local db schema $\mathcal{D}_S$
  - $Materials(mat\_code, cost)$
  - $MO(id, prod\_code, price, status)$
- Then, $\mathcal{D} = \{Materials, Products, PO, MO\}$.
- Parametric actions can introduce values from an infinite domain $U$.
  - $createPO(prod\_code, offer)$ belongs to $Act_C$.
  - $createMO(prod\_code, price)$ belongs to $Act_M$. 


Agents are modules that can be composed together to obtain AC-MAS.

- **Global states** are tuples \( s = \langle D_0, \ldots, D_n \rangle \in \mathcal{D}(U) \).
- An **AC-MAS** is a tuple \( \mathcal{P} = \langle \text{Ag}, s_0, \tau \rangle \) where:
  - \( \text{Ag} = \{0, \ldots, n\} \) is a **finite set of agents**
  - \( s_0 \in \mathcal{D}(U) \) is the **initial global state**
  - \( \tau : \mathcal{D}(U) \times \text{Act}(U) \mapsto 2^{\mathcal{D}(U)} \) is the **transition function**

- **Temporal transition**: \( s \rightarrow s' \) iff there is \( \alpha(\vec{u}) \) s.t. \( s' \in \tau(s, \alpha(\vec{u})) \).
- **Epistemic relation**: \( s \sim_i s' \) iff \( D_i = D'_i \).

AC-MAS are infinite-state systems in general.

AC-MAS are first-order temporal epistemic structures. Hence, FO-CTLK can be used as a specification language.
Syntax: FO-CTLK

- Data call for First-order Logic.
- Evolution calls for Temporal Logic.
- Agents (operating on artifacts) call for Epistemic Logic.

The specification language FO-CTLK:

\[ \varphi ::= P(t) \mid t = t' \mid \neg \varphi \mid \varphi \rightarrow \varphi \mid \forall x \varphi \mid AX \varphi \mid A\varphi U\varphi \mid E\varphi U\varphi \mid Ki\varphi \]

Alternation of free variables and modal operators is enabled.
An AC-MAS $\mathcal{P}$ satisfies an FO-CTLK-formula $\varphi$ in a state $s$ for an assignment $\sigma$, iff

\[
\begin{align*}
(\mathcal{P}, s, \sigma) &\models P_i(t) \quad \text{iff} \quad \langle \sigma(t_1), \ldots, \sigma(t_a) \rangle \in D_s(P_i) \\
(\mathcal{P}, s, \sigma) &\models t = t' \quad \text{iff} \quad \sigma(t) = \sigma(t') \\
(\mathcal{P}, s, \sigma) &\models \neg \varphi \quad \text{iff} \quad (\mathcal{P}, s, \sigma) \not\models \varphi \\
(\mathcal{P}, s, \sigma) &\models \varphi \rightarrow \psi \quad \text{iff} \quad (\mathcal{P}, s, \sigma) \not\models \varphi \text{ or } (\mathcal{P}, s, \sigma) \models \psi \\
(\mathcal{P}, s, \sigma) &\models \forall x \varphi \quad \text{iff} \quad \text{for all } u \in \text{adom}(s), (\mathcal{P}, s, \sigma^x_u) \models \varphi \\
(\mathcal{P}, s, \sigma) &\models AX \varphi \quad \text{iff} \quad \text{for all runs } r, r^0 = s \text{ implies } (\mathcal{P}, r^1, \sigma) \models \varphi \\
(\mathcal{P}, s, \sigma) &\models A\varphi U \varphi' \quad \text{iff} \quad \text{for all runs } r, r^0 = s \text{ implies } (\mathcal{P}, r^k, \sigma) \models \varphi' \text{ for some } k \geq 0, \\
&\quad \text{and } (\mathcal{P}, r^{k'}, \sigma) \models \varphi \text{ for all } 0 \leq k' < k \\
(\mathcal{P}, s, \sigma) &\models E \varphi U \varphi' \quad \text{iff} \quad \text{there exists } r \text{ s.t. } r^0 = s, (\mathcal{P}, r^k, \sigma) \models \varphi' \text{ for some } k \geq 0, \\
&\quad \text{and } (\mathcal{P}, r^{k'}, \sigma) \models \varphi \text{ for all } 0 \leq k' < k \\
(\mathcal{P}, s, \sigma) &\models Ki \varphi \quad \text{iff} \quad \text{for all states } s', s \sim_i s' \text{ implies } (\mathcal{P}, s', \sigma) \models \varphi
\end{align*}
\]

- **Active-domain semantics for quantifiers.**
Semantics of FO-CTLK

Intuition

(d) $AX\varphi$

(e) $A\varphi U\psi$

(f) $E\varphi U\psi$
Verification of AC-MAS

How do we verify FO-CTLK specifications on AC-MAS?

- the manufacturer M knows that each MO has to match a corresponding PO:
  \[ AG \forall id, pc (\exists pr, s \ MO(id, pc, pr, s) \rightarrow K_M \exists o, s' \ PO(id, pc, o, s')) \]

- the client C knows that every PO will eventually be discharged (by M):
  \[ AG \forall id, pc (\exists pr, s \ MO(id, pc, pr, s) \rightarrow EF K_C \exists o \ PO(id, ps, o, shipped)) \]

**Problem:** the infinite domain \( U \) may generate infinitely many states!

**Investigated solution:** can we *simulate* the concrete values from \( U \) with a finite set of *abstract* symbols?
Abstraction: Isomorphism and Bisimulation

- Two states $s, s'$ are **isomorphic**, or $s \simeq s'$, if there is a bijection

  $$\iota : \text{adom}(s) \cup C \leftrightarrow \text{adom}(s') \cup C$$

  such that

  - $\iota$ is the identity on $C$
  - for every $\bar{u} \in \text{adom}(s)^{a_i}, i \in \text{Ag}, \bar{u} \in D_i(P_j) \iff \iota(\bar{u}) \in D_i'(P_j)$

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>2</td>
</tr>
<tr>
<td>$c$</td>
<td>2</td>
</tr>
<tr>
<td>$d$</td>
<td>4</td>
</tr>
<tr>
<td>$e$</td>
<td>5</td>
</tr>
</tbody>
</table>

  $\iota : a \mapsto 1$
  $b \mapsto 2$
  $c \mapsto c$
  $d \mapsto 4$
  $e \mapsto 5$
Abstraction: Isomorphism and Bisimulation

- Two states \(s, s'\) are \textit{bisimilar}, or \(s \approx s'\), if
  - \(s \approx s'\)
  - if \(s \rightarrow t\) then there is \(t'\) s.t. \(s' \rightarrow t'\), \(s \oplus t \approx s' \oplus t'\), and \(t \approx t'\)
Abstraction: Isomorphism and Bisimulation

- Two states $s, s'$ are *bisimilar*, or $s \approx s'$, if
  - $s \approx s'$
  - if $s \rightarrow t$ then there is $t'$ s.t. $s' \rightarrow t'$, $s \oplus t \approx s' \oplus t'$, and $t \approx t'$

  ![Diagram](image)

  - the other direction holds as well
  - similarly for the epistemic relation $\sim_i$
However, bisimulation is not sufficient to preserve FO-CTLK formulas:

\[ \phi = AG \forall x (P(x) \rightarrow AX AG \neg P(x)) \]
Uniformity

- An AC-MAS $\mathcal{P}$ is \textit{uniform} iff for $s, t, s' \in S$ and $t' \in \mathcal{D}(U)$:
  - $s \rightarrow t$ and $s \oplus t \simeq s' \oplus t'$ imply $s' \rightarrow t'$

\begin{itemize}
  \item Intuitively, the behaviour of uniform AC-MAS is independent from data not explicitly named in the system description.
  \item Uniform AC-MAS cover a vast number of interesting cases [2, 4].
\end{itemize}
Uniformity

• An AC-MAS $P$ is *uniform* iff for $s, t, s' \in S$ and $t' \in \mathcal{D}(U)$:
  
  ▶ $s \rightarrow t$ and $s \oplus t \simeq s' \oplus t'$ imply $s' \rightarrow t'$

\[
\begin{array}{|c|c|}
\hline
s & t \\
\hline
a & b \\
\hline
b & c \\
\hline
d & e \\
\hline
\end{array}
\quad \rightarrow \quad
\begin{array}{|c|c|}
\hline
t & \\
\hline
a & f \\
\hline
f & c \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
s' & t' \\
\hline
1 & 2 \\
\hline
2 & c \\
\hline
4 & 5 \\
\hline
\end{array}
\quad \rightarrow \quad
\begin{array}{|c|c|}
\hline
t' & \\
\hline
1 & 6 \\
\hline
6 & c \\
\hline
\end{array}
\]

• Intuitively, the behaviour of uniform AC-MAS is independent from data not explicitly named in the system description.

• Uniform AC-MAS cover a vast number of interesting cases [2, 4].
Bisimulation and Equivalence w.r.t. FO-CTLK

Theorem

Consider

- bisimilar and uniform AC-MAS $\mathcal{P}_1$ and $\mathcal{P}_2$
- an FO-CTLK formula $\varphi$

If

1. $|U_2| \geq 2 \cdot \sup_{s \in \mathcal{P}_1} |\text{adom}(s)| + |C| + |\text{vars}(\varphi)|$
2. $|U_1| \geq 2 \cdot \sup_{s' \in \mathcal{P}_2} |\text{adom}(s')| + |C| + |\text{vars}(\varphi)|$

then

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

Can we apply this result to finite abstraction?
Abstractions

• Abstractions are defined in an agent-based, modular manner.

• Let \( A = \langle \mathcal{D}, Act, Pr \rangle \) be an agent defined on the domain \( U \).

  Given a domain \( U' \), the abstract agent \( A' = \langle \mathcal{D}', Act', Pr' \rangle \) on \( U' \) is s. t.
  ◀ \( \mathcal{D}' = \mathcal{D} \)
  ◀ \( Act' = Act \)
  ◀ \( Pr' \) is the smallest function s. t. if \( \alpha(\vec{u}) \in Pr(D) \), \( D' \in \mathcal{D}'(U') \) and \( D' \simeq D \) for some witness \( \iota \), then \( \alpha(\vec{u}') \in Pr'(D') \) where \( \vec{u}' = \iota'(\vec{u}) \) for some constant-preserving bijection \( \iota' \) extending \( \iota \) to \( \vec{u} \).

• Given a set \( Ag \) of agents on \( U \), let \( Ag' \) be the set of abstract agents on \( U' \).

• Let \( P = \langle Ag, s_0, \tau \rangle \) be an AC-MAS. The AC-MAS \( P' = \langle Ag', s'_0, \tau' \rangle \) is an abstraction of \( P \) iff
  ◀ \( s'_0 = s_0 \);
  ◀ \( \tau' \) is the smallest function s. t. if \( t \in \tau(s, \alpha(\vec{u})) \), \( s', t' \in D'(U') \) and \( s \oplus t \simeq s' \oplus t' \), for some witness \( \iota \), then \( t' \in \tau'(s', \alpha(\vec{u}')) \), where \( \vec{u}' = \iota'(\vec{u}) \) for some constant-preserving bijection \( \iota' \) extending \( \iota \) to \( \vec{u} \).
Bounded Models and Finite Abstractions

• An AC-MAS $\mathcal{P}$ is \textit{b-bounded} iff for all $s \in \mathcal{P}$, $|\text{dom}(s)| \leq b$.
• Bounded systems can still be infinite!

**Theorem**

Consider

• a $b$-bounded and uniform AC-MAS $\mathcal{P}$ on an infinite domain $U$
• an FO-CTLK formula $\varphi$.

Given $U' \supseteq C$ s.t.

$$|U'| \geq 2b + |C| + \max\{|\text{vars}(\varphi)|, N_{Ag}\}$$

there exists a finite abstraction $\mathcal{P}'$ of $\mathcal{P}$ s.t.

• $\mathcal{P}'$ is uniform and bisimilar to $\mathcal{P}$

In particular,

$$\mathcal{P} \models \varphi \iff \mathcal{P}' \models \varphi$$

How can we define finite abstractions constructively?
Extensions

1. Non-uniform AC-MAS: for sentence-atomic FO-CTL the results above still hold.

\[ \forall c (\text{shippedPO}(c) \rightarrow \forall m(\text{related}(c, m) \rightarrow \text{shippedMO}(m))) \]

2. Non-uniform AC-MAS: one-way preservation result for FO-ACTL.

---

**Theorem**

*If an AC-MAS $\mathcal{P}$ is bounded, and $\varphi \in \text{FO-ACTL}$, then there exists a finite abstraction $\mathcal{P}'$ such that if $\mathcal{P}' \models \varphi$ then $\mathcal{P} \models \varphi$.***

3. Model checking bounded AC-MAS w.r.t. FO-CTL is undecidable.

4. Complexity result:

**Theorem**

*The model checking problem for finite AC-MAS w.r.t. FO-CTLK is EXPSPACE-complete in the size of the formula and data.*

5. The finite abstraction result can be extended to typed FO-CTLK including predicates with an infinite interpretation ($<$ on rationals)
Results
and main limitations

- We are able to model check AC-MAS w.r.t. full FO-CTLK...
- ...however, our results hold only for uniform and bounded systems.
- This class includes many interesting systems (AS programs, [2, 4]).
- The model checking problem is EXPSPACE-complete.
Next Steps

- Techniques for finite abstraction.
- Model checking techniques for finite-state systems are effective on the abstract system?
- How to perform the boundedness check.
Merci!
Christel Baier and Joost-Pieter Katoen.  
*Principles of Model Checking.*  

D. Cohn and R. Hull.  

*Reasoning About Knowledge.*  

B. Bagheri Hariri, D. Calvanese, G. De Giacomo, R. De Masellis, and P. Felli.  
Foundations of Relational Artifacts Verification.  