Verification of Artifact-Centric Multi-Agent Systems via Finite Abstraction: Some Decidability Results

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)

LACL – 17 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

(a) E. Clarke (CMU, USA)
(b) A. Emerson (U. Texas, USA)
(c) J. Sifakis (IMAG, F)

• Jury justification

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

Motivation: Artifact Systems as data-aware systems
Overview

1 Motivation: Artifact Systems as data-aware systems

2 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].
Overview

1 Motivation: Artifact Systems as *data-aware* systems

2 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].

3 Key contribution: Verification of *bounded* and *uniform* AS is decidable
• 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: interacting artifacts, representing services, manipulated by agents.
Artifact 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>
## Artifact Systems

### Data Model

#### PO

<table>
<thead>
<tr>
<th>id</th>
<th>prod_code</th>
<th>offer</th>
<th>status</th>
</tr>
</thead>
</table>

- `createPO(prod_code, offer)`
- `deletePO(id)`
- `addItemPO(id, itm, qty)`
- ...

#### MO

<table>
<thead>
<tr>
<th>id</th>
<th>prod_code</th>
<th>price</th>
<th>status</th>
</tr>
</thead>
</table>

- `createMO(id, price)`
- `deleteMO(id)`
- `addLineItemMO(id, mat, qty)`
- ...

Artifact Systems

Lifecycle

- Agents operate on artifacts.
  - e.g., the Customer sends the Purchase Order to the Manufacturer.
- Actions add/remove artifacts or change artifact attributes.
  - e.g., the PO status changes from *created* to *submitted*.
- The whole system can be seen as a *data-aware* dynamic system.
  - at every step, an action yields a change in the current state.
Research questions

1. Which syntax and semantics 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?
Research questions

1. Which syntax and semantics to specify AS?
2. Is verification of AS decidable?
Research questions

1. Which syntax and semantics to specify AS?
2. Is verification of AS decidable?
3. If not, can we identify relevant fragments that are reasonably well-behaved?
Research questions

1. Which syntax and semantics 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 . . .
Challenges

Multi-agent systems, but . . .

- . . . states have a relational structure,
Challenges

Multi-agent systems, but . . .

- . . . states have a relational structure,
- data are potentially infinite,
Challenges

Multi-agent systems, but . . .

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

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

   the manufacturer M knows that each MO has to match a corresponding PO.
Artifact-centric multi-agent systems (AC-MAS): formal model for AS.

Intuition: databases that evolve in time and are manipulated by agents.

FO-CTLK as a specification language:

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

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

Abstraction techniques and finite interpretation to tackle model checking.

Main result: under specific conditions MC can be reduced to the finite case.
Artifact Systems

Results

   **Intuition:** databases that evolve in time and are manipulated by agents.

2. FO-CTLK as a specification language:
   
   \[ AG \forall id, pc (\exists \vec{x} \text{ MO}(id, pc, \vec{x}) \rightarrow K_M \exists \vec{y} \text{ PO}(id, pc, \vec{y})) \]

   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.

4. Modelling of declarative GSM systems, developed by IBM, as AC-MAS.
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 $U$.
- **Disjoint union**: $D \oplus D'$ is the $(\mathcal{D} \cup \mathcal{D}')$-interpretation s.t.
  - (i) $D \oplus D'(P_i) = D(P_i)$
  - (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 \( \vec{x} \)
  - \( Pr_i : D_i(U) \rightarrow 2^{Act_i(U)} \) 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 \( \mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_n \).
Example 1: the Order-to-Cash Scenario

- **Agents:** Customer, Manufacturer, Supplier.
- **Local db schema $D_C$**
  - $Products$(prod code, budget)
  - $PO$(id, prod code, offer, status)
- **Local db schema $D_M$**
  - $PO$(id, prod code, offer, status)
  - $MO$(id, prod code, price, status)
- **Local db schema $D_S$**
  - $Materials$(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 $Act_M$. 
Artifact-centric Multi-agent Systems
AC-MAS

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:

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

Alternation of free variables and modal operators is enabled.
Semantics of FO-CTLK

Formal definition

An AC-MAS $\mathcal{P}$ satisfies an FO-CTLK-formula $\varphi$ in a state $s$ for an assignment $\sigma$, iff

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

- **Active-domain semantics**: $\text{adom}(D)$ is the set of all $u \in U$ appearing in $D$
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 $PO$:
  
  \[ \text{AG } \forall \text{id}, \text{pc} (\exists \text{pr}, s \, \text{MO}(\text{id}, \text{pc}, \text{pr}, s) \rightarrow K_M \exists o, s' \, \text{PO}(\text{id}, \text{pc}, o, s')) \]

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

**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 \textit{isomorphic}, or \( s \simeq s' \), if there is a bijection

\[
\iota : adom(s) \cup C \mapsto adom(s') \cup C
\]

such that

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

\[
\begin{array}{c|c|c}
D & D' \\
\hline
a & b & 1 \\
\hline
b & c & 2 \\
\hline
d & e & c \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
D' & 1 & 2 \\
\hline
2 & c & 4 \\
\hline
4 & 5 & 5 \\
\hline
\end{array}
\]

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

- Two states $s, s'$ are *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](attachment:image.png)
Two states $s, s'$ are \textit{bisimilar}, or $s \approx s'$, if

\begin{itemize}
  \item $s \approx s'$
  \item if $s \rightarrow t$ then there is $t'$ s.t. $s' \rightarrow t'$, $s \oplus t \approx s' \oplus t'$, and $t \approx t'$
\end{itemize}

\begin{align*}
  s & \rightarrow t \\
  \approx & \\
  s' & \rightarrow t'
\end{align*}

\begin{itemize}
  \item the other direction holds as well
  \item similarly for the epistemic relation $\sim_i$
\end{itemize}
Abstraction: Isomorphism and Bisimulation

However, bisimulation is not sufficient to preserve FO-CTLK formulas:

\[
\phi = AG \forall x (P(x) \rightarrow AX AG \neg P(x))
\]
Uniform AC-MAS cover a vast number of interesting cases [2, 4].

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

- Intuitively, the behaviour of uniform AC-MAS is independent from data not explicitly named in the system description.
- More formally, an AC-MAS $\mathcal{P}$ 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 $s' \rightarrow t'$

\[
\begin{array}{cc}
s & t \\
\hline
a & b \\
b & c \\
d & e \\
\end{array}
\]

\[
\begin{array}{cc}
s' & t' \\
\hline
1 & 2 \\
2 & c \\
4 & 5 \\
\end{array}
\]
Uniformity

- Intuitively, the behaviour of uniform AC-MAS is independent from data not explicitly named in the system description.
- More formally, an AC-MAS $\mathcal{P}$ 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 $s' \rightarrow t'$

  ![Uniformity Diagram]

- 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 way.
- Let $A = \langle D, Act, Pr \rangle$ be an agent defined on the domain $U$.
  Given a domain $U'$, the abstract agent $A' = \langle D', Act', Pr' \rangle$ on $U'$ is s.t.
  - $D' = D$
  - $Act' = Act$
  - $Pr'$ is the smallest function s.t. if $\alpha(\bar{u}) \in Pr(D)$, $D' \in D'(U')$ and $D' \simeq D$ for some witness $\iota$, then $\alpha(\bar{u}') \in Pr'(D')$ where $\bar{u}' = \iota'((\bar{u})$ for some constant-preserving bijection $\iota'$ extending $\iota$ to $\bar{u}$.
- 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
  - $s'_0 \simeq s_0$;
  - $\tau'$ is the smallest function s.t. if $t \in \tau(s, \alpha(\bar{u}))$, $s', t' \in D'(U')$ and $s \oplus t \simeq s' \oplus t'$ for some witness $\iota$, then $t' \in \tau'(s', \alpha(\bar{u}'))$ where $\bar{u}' = \iota'(\bar{u})$ for some constant-preserving bijection $\iota'$ extending $\iota$ to $\bar{u}$. 

25
Bounded Models and Finite Abstractions

- An AC-MAS $\mathcal{P}$ is $b$-bounded iff for all $s \in \mathcal{P}$, $|\text{adom}(s)| \leq b$.
- 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.

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

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?
Compact descriptions: AS Programs

Example of uniform AC-MAS written in a FO language.

- for each agent $i$, $\text{Act}_i$ is the set of of local (parametric) actions of the form $\omega(\vec{x}) = \langle \pi(\vec{y}) , \psi(\vec{z}) \rangle$ s.t.
  - $\omega(\vec{x})$ is the operation signature and $\vec{x} = \vec{y} \cup \vec{z}$ is the set of operation parameters
  - $\pi(\vec{y})$ is the operation precondition, i.e., an FO-formula over $D_i$
  - $\psi(\vec{z})$ is the operation postcondition, i.e., an FO-formula over $D \cup D'$

We call the AC-MAS specified in this way Artifact System Programs.
Example 2: the Order-to-Cash Scenario

Specification of actions affecting the MO in the order-to-cash scenario:

- $createMO(po_id, \text{price}) = \langle \pi(po_id, \text{price}), \psi(po_id, \text{price}) \rangle$, where:
  - $\pi(po_id, \text{price}) \equiv \exists p, o \ (PO(po_id, p, o, \text{prepared}) \land \exists \text{cost} \ Materials(p, \text{cost}) \land \phi_{b-1}$
  - $\psi(po_id, \text{price}) \equiv \exists id \ (MO'(id, po_id, \text{price}, \text{preparation}) \land \forall id', c, p, s \ (MO(id', c, p, s) \rightarrow id \neq id')) \land \phi_b$

where $\phi_k$ is the FO-formula saying that there are at most $k$ objects in the active domain.

The specification of $createMO$ guarantees that the bound $b$ is not violated by action execution.
Verification of Artifact System Programs

Lemma

AS programs generate uniform AC-MAS.

Theorem

Consider

- a $b$-bounded AS program $\mathcal{P}_{\text{Act},U}$ on an infinite domain $U$
- an FO-CTLK formula $\varphi$.

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

$$|U_2| \geq 2b + |C| + \max\{N_{\text{AS}}, |\text{vars}(\varphi)|\}$$

then $\mathcal{P}_{\text{Act},U'}$ is a finite abstraction of $\mathcal{P}_{\text{Act},U}$ s.t.

- $\mathcal{P}_{\text{Act},U'}$ is uniform and bisimilar to $\mathcal{P}_{\text{Act},U}$

In particular,

$$\mathcal{P}_{\text{Act},U} \models \varphi \text{ iff } \mathcal{P}_{\text{Act},U'} \models \varphi$$

- The abstraction is finite and the procedure is constructive.
- Thus, we can apply standard techniques in model checking.
Extensions

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

\[ AG \forall c ( shippedPO(c) \rightarrow \forall m( related(c, m) \rightarrow shippedMO(m))) \]
Extensions

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

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

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

**Theorem**

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

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

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

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.
Extensions

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

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

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

Theorem

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

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

- 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.*
Extensions

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

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

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

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

4. Complexity result:

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 \( \mathcal{P} \models \varphi \).

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.