Verification of Agent-based Artifact Systems: Abstraction Techniques and 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)

Laboratoire LIP6 – 25 February 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
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.
Artifact Systems
Order-to-Cash Scenario

Customer → Purchase Order
Desk 1, Chair 4

Manufacturer → Material Order
Desk Legs 4, Chair Legs 16,
... ...

Supplier → Accept/reject

Material Order
Hammer Nails 1000, Glue 10,
... ...

Accept/reject
Artifact Systems

Data Model

<table>
<thead>
<tr>
<th>PO</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>prod_code</td>
<td>offer</td>
<td>status</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>MO</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>prod_code</td>
<td>price</td>
<td>status</td>
<td></td>
</tr>
</tbody>
</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 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?
Research questions

1. Which syntax and semantics should we use to specify AS?
Research questions

1. Which syntax and semantics should we use to specify AS?
2. Is verification of AS decidable?
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?
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 . . .
Challenges

Multi-agent systems, but . . .

• . . . states have a relational structure,
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.

2. 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.

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.
Semantics: Databases

The data model of Artifact Systems is given as a database.

- a **database schema** is a finite set \( D = \{ P_1/a_1, \ldots, P_n/a_n \} \) of predicate symbols \( P_i \) with arity \( a_i \in \mathbb{N} \).

- a **\( D \)-interpretation** 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 \( D \)

- the **primed version** of the db schema \( D \) as above is the db schema \( D' = \{ P'_1/a_1, \ldots, P'_n/a_n \} \).

- **Composition**: \( D \oplus D' \) is the \((D \cup D')\)-interpretation s.t.

  (i) \( D \oplus D'(P_i) = D(P_i) \), and

  (ii) \( D \oplus D'(P'_i) = D'(P_i) \).
Artifact-centric Multi-agent Systems

Agents

Agents have partial access (views) to the artifact system.

• an agent is a tuple $i = \langle D_i, L_i, Act_i, Pr_i \rangle$ where
  ▶ $D_i$ is the local database schema
  ▶ $L_i \subseteq D_i(U)$ is the set of local states
  ▶ $Act_i$ is the set of local actions $\alpha(\vec{x})$ with parameters $\vec{x}$
  ▶ $Pr_i : L_i \mapsto 2^{\text{Act}_i}$ is the local protocol function

• the global database schema is defined as $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_n$.

• 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 $\mathcal{D}$. 
Example 1: the Order-to-Cash Scenario

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

AC-MAS

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

• An **AC-MAS** is a tuple \( \mathcal{P} = \langle S, U, D_0, \tau \rangle \) where:
  - \( S \subseteq L_1 \times \cdots \times L_n \) is the set of *reachable global states*
  - \( U \) is the *interpretation domain*
  - \( D_0 \in S \) is the *initial global state*
  - \( \tau : S \times \text{Act}(U) \mapsto 2^S \) is the *transition function*

• **Temporal transition**: \( D \rightarrow D' \) iff there is \( \alpha(\bar{u}) \) s.t. \( D' \in \tau(D, \alpha(\bar{u})) \).

• **Epistemic relation**: \( D \sim_i D' \) 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 variables and path quantifiers is enabled.
Semantics of FO-CTLK

Formal definition

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

$$(\mathcal{P}, D, \sigma) \models P_i(\bar{t}) \quad \text{iff} \quad \langle \sigma(t_1), \ldots, \sigma(t_\ell) \rangle \in D(P_i)$$

$$(\mathcal{P}, D, \sigma) \models t = t' \quad \text{iff} \quad \sigma(t) = \sigma(t')$$

$$(\mathcal{P}, D, \sigma) \models \neg \varphi \quad \text{iff} \quad (\mathcal{P}, D, \sigma) \not\models \varphi$$

$$(\mathcal{P}, D, \sigma) \models \varphi \rightarrow \psi \quad \text{iff} \quad (\mathcal{P}, D, \sigma) \not\models \varphi \text{ or } (\mathcal{P}, D, \sigma) \models \psi$$

$$(\mathcal{P}, D, \sigma) \models \forall x \varphi \quad \text{iff} \quad \text{for all } u \in \text{adom}(D), (\mathcal{P}, D, \sigma^u) \models \varphi$$

$$(\mathcal{P}, D, \sigma) \models AX \varphi \quad \text{iff} \quad \text{for all runs } r, r^0 = D \text{ implies } (\mathcal{P}, r^1, \sigma) \models \varphi$$

$$(\mathcal{P}, D, \sigma) \models A\varphi U \varphi' \quad \text{iff} \quad \text{for all runs } r, r^0 = D \text{ implies } (\mathcal{P}, r^k, \sigma) \models \varphi' \text{ for some } k \geq 0,$$

and $(\mathcal{P}, r^{k'}, \sigma) \models \varphi$ for all $0 \leq k' < k$

$$(\mathcal{P}, D, \sigma) \models E\varphi U \varphi' \quad \text{iff} \quad \text{there exists } r \text{ s.t. } r^0 = D, (\mathcal{P}, r^k, \sigma) \models \varphi' \text{ for some } k \geq 0,$$

and $(\mathcal{P}, r^{k'}, \sigma) \models \varphi$ for all $0 \leq k' < k$

$$(\mathcal{P}, D, \sigma) \models K_i \varphi \quad \text{iff} \quad \text{for all runs } r, n \in \mathbb{N}, D \sim_i r^n \text{ implies } (\mathcal{P}, r^n, \sigma) \models \varphi$$

- 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:
  \[ \forall id, pc \ (\exists pr, s \ MO(id, pc, pr, s) \rightarrow K_M \ \exists o, s' \ PO(id, pc, o, s')) \]

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

Problem: the infinite domain \( U \) can determine 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 $D, D'$ are **isomorphic**, or $D \simeq D'$, if there is a bijection $\iota : \text{adom}(D) \cup C \mapsto \text{adom}(D') \cup C$ s.t.
  - $\iota$ is the identity on $C$
  - for every $\bar{u} \in \text{adom}(D)^a_i$, $i \in \text{Ag}$, $\bar{u} \in D_i(P_j) \iff \iota(\bar{u}) \in D'_i(P_j)$

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

```
    D -----> E
    |      |
    |  ≃   |
    |      |
      D'   
```
Abstraction: Isomorphism and Bisimulation

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

```
D -> E

≈
≈

D' -> E'
```

- similarly for the epistemic relation $\sim_i$
- the other direction holds as well
Abstraction: Isomorphism and Bisimulation

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

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

- An AC-MAS $\mathcal{P}$ is *uniform* iff for $D, E, D' \in S$ and $E' \in \mathcal{D}(U)$:
  - $D \to E$ and $D \oplus E \simeq D' \oplus E'$ imply $D' \to E'$

<table>
<thead>
<tr>
<th>$D$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D'$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E'$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Uniformity

• An AC-MAS \( \mathcal{P} \) is uniform iff for \( D, E, D' \in S \) and \( E' \in D(U) \):
  
  ▶ \( D \rightarrow E \) and \( D \oplus E \simeq D' \oplus E' \) imply \( D' \rightarrow E' \)

\[
\begin{array}{c|c}
D & E \\
\hline
a & b \\
b & c \\
d & e \\
\end{array}
\rightarrow
\begin{array}{c|c}
E & \end{array}
\rightarrow
\begin{array}{c|c}
\hline
1 & 2 \\
2 & 3 \\
4 & 5 \\
\end{array}
\rightarrow
\begin{array}{c|c}
E' & \end{array}
\rightarrow
\begin{array}{c|c}
\hline
1 & 6 \\
6 & 3 \\
\end{array}
\]

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

then

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

Can we apply this result to finite abstraction?
Abstractions

• Let \( A = \langle D, L, Act, Pr \rangle \) be an agent defined on the domain \( U \).
  Given a domain \( U' \), the abstract agent \( A' = \langle D', L', Act', Pr' \rangle \) on \( U' \) is s. t.
  \( \triangleright D'_i = D_i \)
  \( \triangleright L'_i = D'_i(U') \)
  \( \triangleright Act'_i = Act_i \)
  \( \triangleright \alpha(\vec{u}') \in Pr'_i(l'_i) \) iff there exist \( l_i \in L_i \) and \( \alpha(\vec{u}) \in Pr_i(l_i) \) s.t. \( l'_i \simeq l_i \), for some witness \( \iota \), and \( \vec{u}' = \iota'(\vec{u}) \), for some 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 \( P = \langle S, U, D_0, \tau \rangle \) be an AC-MAS on the set \( Ag \) of agents.
  The AC-MAS \( P' = \langle S', U', D'_0, \tau' \rangle \) on the set \( Ag' \) of abstract agents is an \( \oplus \)-abstraction of \( P \) iff:
  \( \triangleright D'_0 = D_0; \)
  \( \triangleright t' \in \tau'(s', \vec{\alpha}(\vec{u}')) \) iff there exist \( s, t \in S \) and \( \vec{\alpha}(\vec{u}) \in Act(U) \), such that \( s \oplus t \simeq s' \oplus t' \), for some witness \( \iota \), \( t \in \tau(s, \vec{\alpha}(\vec{u})) \), and \( \vec{u}' = \iota'(\vec{u}) \) for some bijection \( \iota' \) extending \( \iota \) to \( \vec{u} \).
Bounded Models and Finite Abstractions

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

- \(createMO(po_id, price) = \langle \pi(po_id, price), \psi(po_id, price) \rangle\), where:

  - \(\pi(po_id, price) \equiv \exists p, o \ (PO(po_id, p, o, prepared) \land \exists cost \ Materials(p, cost) \land \phi_{b-1}\)
  - \(\psi(po_id, price) \equiv \exists id \ (MO'(id, po_id, price, preparation)\land\)

\[\forall id', c, p, s \ (MO(id', c, p, s) \rightarrow id \neq id')) \land \phi_b\]

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,

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

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

1. Non-uniform AC-MAS: for the sentence-atomic fragment of 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 \( \mathcal{P} \) 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.
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.
- Abstraction 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.  