Verification of Deployed Artifact Systems via Data Abstraction

Fabio Patrizi

Sapienza Università di Roma, Italy
patrizi@dis.uniroma1.it

Joint work with Francesco Belardinelli and Alessio Lomuscio
Imperial College London, UK

RMIT – Melbourne
December 9, 2011
Overview

1. Motivation: Artifact Systems
2. Verification of infinite-state data-aware systems
3. Key contribution: decidability under boundedness assumption
4. Application of the result
5. Conclusion and future directions
Artifact and Artifact Systems

Recent paradigm for Business Process modeling and development [CH09]

- **Artifact**: information model + lifecycle
  - (Nested) records equipped with actions
- **Artifact System**: set of interacting artifacts

Features:

- Data and processes are given same emphasis
  - data affect the actions to execute
  - actions affect data (content and structure)
- Modularized approach (sort of Object-Orientation)
  - focus on one artifact at a time
Artifact Systems

Motivating 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>

Accept/reject
Motivating Scenario (cont.)

<table>
<thead>
<tr>
<th>CPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

- `createPO(id, cid, code)`
- `deletePO(id)`
- `addItemPO(id, itm, qty)`
- `...`

<table>
<thead>
<tr>
<th>WO</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

- `createWO(id, cpo)`
- `deleteWO(id)`
- `addLineItemWO(id, mat, qty)`
- `...`
Artifact Systems

- As the process goes on, artifact actions are executed
  - e.g., the Customer Purchase Order is sent to the Manufacturer.
- Actions add/remove artifacts or change artifact attributes
  - e.g., the CPO 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
Preliminary (standard) notions and notation

- A **database schema** is a set $D = \{P_1/a_1, \ldots, P_n/a_n\}$ of relation symbols $P_i$, each with its *arity* $a_i$.

- A **$D$-interpretation** (or *instance*) over (possibly infinite) $U$ is a mapping associating each $P_i$ with a finite $a_i$-ary relation $D(P_i) \subseteq U^{a_i}$.

- **Active domain**: $\text{adom}(D) \subseteq U$ is the (finite) set of all distinct elements occurring in $D$.

- First-Order formulas/sentences are syntactically defined as usual but evaluated under **active-domain semantics**:
  - *quantified variables range over the active domain*.
How do we describe an Artifact System?

**Definition (Artifact System)**

An *Artifact System* is specified as a tuple $S = \langle D, U, D_0, \Phi \rangle$, where:

- $D = \{P_1/a_1, \ldots, P_n/a_n\}$ is a *database schema*
- $U$ is a possibly infinite *interpretation domain*
- $D_0$ is an *initial* $D$-instance over $U$
- $\Phi$ is a finite set of *parametric actions* of the form $\alpha(\vec{x}) = \langle \pi(\vec{y}), \psi(\vec{z}) \rangle$, where:
  - $\alpha(\vec{x})$ is the *action signature* and $\vec{x}$ the set of its *formal parameters*
  - $\vec{x} = \vec{y} \cup \vec{z}$
  - $\pi(\vec{y})$ is a FO-formula over $D$ called the *action precondition*
  - $\psi(\vec{z})$ is a FO-formula over $D \cup D'$ called the *action postcondition*, where $D' \doteq \{P'_1/a_1, \ldots, P'_n/a_n\}$
Framework
Artifact Systems: Semantics

Definition (Model of an Artifact System)

Given an Artifact System $S = \langle D, U, D_0, \Phi \rangle$, its model is the Kripke structure $K = \langle \Sigma, D_0, \tau \rangle$, where:

- $\Sigma \subseteq \mathcal{I}_D(U)$ is the set of states ($\mathcal{I}_D(U)$: all instances of $D$ over $U$)
- $D_0 \in \Sigma$ is the initial state
- $\tau : \Sigma \to \Sigma$ is the transition relation s.t. $\tau(D, D')$ iff for some $\alpha$ there exists an execution $\alpha(\vec{u}) = \langle \pi(\vec{v}), \psi(\vec{w}) \rangle$ such that:
  - $\text{adom}(D') \subseteq \text{adom}(D) \cup \{w_1, \ldots, w_\ell\} \cup \text{const}(\psi)$
  - $D \models \pi(\vec{v})$, i.e., the action is enabled
  - $D \uplus D' \models \psi(\vec{w})$, where $D \uplus D'$ interprets unprimed symbols as in $D$ and primed ones as in $D'$.

NOTE: First-Order formulas evaluated under active-domain semantics.
Framework

Intuition

- Each state is a $\mathcal{D}$-instance
- As actions are executed, new states are reached
- Action parameters can introduce new values
- Infinite $U$ yields potentially infinitely many distinct states
- In general, infinite branching and infinite run-length
The Problem

Intuition

Check whether all possible system evolutions satisfy a desired property

- Does the system satisfy a (branching-time) temporal specification? E.g.:
  - It is always the case that every artifact can be deleted
  - There exists a way to create a certain number of artifacts
  - A product can be shipped to the customer only after assemblage

- Flavor of Model Checking, but:

Relational states + infinite interpretation domain = infinite state space!
Verification Formalism: FO-CTL

Syntax

How to specify system properties?

Definition (Syntax of FO-CTL over $S$)

$$\varphi ::= \phi \mid \varphi \land \varphi \mid \neg \varphi \mid AX \varphi \mid \spU \varphi \mid E \spU \varphi,$$

where $\phi$ is a FO-sentence over $D$ and $U$.

(Other operators derived as usual)

Essentially, CTL with propositional formulas replaced by FO sentences

E.g.:

- $\varphi_{ship} = AG \forall c \left( shippedCPO(c) \rightarrow \forall m \left( related(c, m) \rightarrow shippedMPO(m) \right) \right)$

- $\varphi_{t+} = EF \exists x_1, \ldots, x_{t+1} \land_{i \neq j} x_i \neq x_j$

- $\varphi_{empty} = AG EF \left( emptyCPO \land emptyWO \land emptyMPO \right)$
Verification Formalism: FO-CTL

Semantics

(A run r is a sequence of successor states. r(i) selects the i-th r-state.)

Definition (Semantics of FO-CTL over S)

Let K be the model of S and D ∈ Σ a K-state.

(K, D) |= ϕ iff D |= ϕ, if ϕ is an FO-sentence;
(K, D) |= ¬ϕ iff (K, D) \not|= ϕ;
(K, D) |= ϕ → ψ iff (K, D) \not|= ϕ or (K, D) |= ψ;
(K, D) |= AXϕ iff for all K-runs r s.t. r(0) = D, (K, r(1)) |= ϕ;
(K, D) |= AϕUψ iff for all K-runs r s.t. r(0) = D, ∃k ≥ 0 s.t. (K, r(k)) |= ψ
and ∀j s.t. 0 ≤ j < k, (K, r(j)) |= ϕ;

(K, D) |= EϕUψ iff for some K-run r, r(0) = D, ∃k ≥ 0 s.t. (K, r(k)) |= ψ,
and ∀j s.t. 0 ≤ j < k, (K, r(j)) |= ϕ.

A formula ϕ is true in K, written K |= ϕ, if (K, D_0) |= ϕ.

S satisfies ϕ, written S |= ϕ, if K |= ϕ.
FO-CTL Semantics

Intuition

$AX \varphi$: 

$A\varphi U \psi$: 

$E\varphi U \psi$: 
Verification of Artifact Systems

General Formulation

- **Model Checking problem for Artifact Systems:**
  
  Given $S$ and $\varphi$, does $S \models \varphi$ hold?

- Similar to Model Checking but technically more challenging
  - Relational states
  - Infinite state-space

**Theorem**

*The MC problem for Artifact Systems is undecidable.*

- BUT decidable over finite interpretation domains:
  - by reduction to standard propositional case (*propositionalise* FO facts).
Verification of Bounded Artifact Systems

- Here we devise a notable case of decidability
- If all the $D$-instances (states) of the system are bounded, then, though infinite-state, model-checking the system is decidable.
Bounded Artifact System

Definition (\(b\)-Bounded (Artifact) System)

Consider a system \( S = \langle D, U, D_0, \Phi \rangle \), and a bound \( b \in \mathbb{N} \) such that \( b \geq |D_0| \). \( S \) is \( b\)-bounded if its model \( K_b = \langle \Sigma_b, D_0, \tau_b \rangle \) is such that

- for every \( D \in \Sigma_b, |D| \leq b \)
Verification of Bounded Artifact Systems

We consider the following problem:

\* Model Checking of Bounded Artifact Systems:

Given a b-bounded artifact system $S$ and a property $\varphi$, does $K_b \models \varphi$?
Verification of Bounded Artifact Systems

Cont.

As a result of the infinite interpretation domain, we still have:

- Infinite branching
- Infinite state-space

QUESTIONS:

- Is the problem decidable?
- 🕵️‍♀️ How can we model-check a bounded system?

Non-trivial! (we cannot *construct* the (infinite) model)
Abstract System

Definition

Given a $b$-bounded system $S = \langle D, U, D_0, \Phi \rangle$ and a property $\varphi$, the $(b, \varphi)$-bounded Abstract System of $S$ is the Artifact System $\hat{S}_{b, \varphi} = \langle D, \hat{U}, D_0, \Phi \rangle$, s.t. $\hat{U} = C_{S, \varphi} \cup \hat{C}$, with:

- $C_{S, \varphi} = const(\varphi) \cup \bigcup_{\phi \in \Phi} const(\phi)$
- $\hat{C} \cap C_{S, \varphi} = \emptyset$
- $|\hat{C}| = b + v$, with $v = \max_{\phi \in \Phi} \{|vars(\phi)|\}$

Intuition:

- $\hat{S}_{b, \varphi}$ analogous to $S$ except for $U \neq \hat{U}$
- $\hat{U}$ contains:
  - all constants mentioned in $S$ and $\varphi$
  - enough distinct abstract symbols to “fill” the bound and have “fresh” actual parameters for action executions
- $\hat{U}$ is finite!
Abstract System Verification

- Obviously, $\hat{S}_{b,\varphi} \models \varphi$ is decidable, as $\hat{U}$ is finite
- But we want to check whether $\mathcal{K}_{b} \models \varphi$
- So, what is the relationship between $\mathcal{K}_{b}$ and $\hat{S}_{b,\varphi}$?

Theorem

Consider a $b$-bounded system $S$ with $U$ infinite, and a FO-CTL specification $\varphi$.\textsuperscript{a} If $\hat{S}_{b,\varphi}$ is the $(b, \varphi)$-bounded abstract system of $S$ then

$$\mathcal{K}_{b} \models \varphi \Leftrightarrow \hat{\mathcal{K}}_{b,\varphi} \models \varphi,$$

where:

- $\mathcal{K}_{b}$ is the model of $S$, and
- $\hat{\mathcal{K}}_{b,\varphi}$ is the model of $\hat{S}_{b,\varphi}$.

\textsuperscript{a}In fact for the whole FO $\mu$-calc
Complexity

- Upper bound:
  \[ O(2|\hat{U}|^a + |\hat{U}|^{|\varphi|}) \]

- Technique based on reduction to propositional CTL MC (viable as abstract interpretation domain finite)
  \[ \hat{K}_{b,\varphi} \text{-states propositionalised (single exponential wrt } |\hat{U}| = b + v + |C_{\delta,\varphi}| \text{ but doubly wrt } a) \]

- Quantifiers eliminated from \( \varphi \) (single exponential in \( |\varphi| \))

Observations:
- Not far from similar results ([DSV07, DHPV09, BCD¹¹])
- Some performing well in practice ([DSV07])
- Non-optimal technique
Abstract System Verification

Technique

$\hat{\mathcal{K}}_{b, \varphi} \models \varphi$ can be reduced to standard MC

We have an actual technique to model-check $\mathcal{K}_b$!
Data Abstraction

\[ \mathcal{K}_b \models \varphi \iff \hat{\mathcal{K}}_{b,\varphi} \models \varphi \]

- What’s behind the scene?
- How did we get rid of an infinite number of elements and transitions?

We applied an *abstraction process* based on two formal notions:

1. **Isomorphism** between DB instances
2. **Bisimulation** between Kripke structures
Definition (C-isomorphic $\mathcal{D}$-instances)

Two $\mathcal{D}$-instances $D$ and $\hat{D}$, respectively over $U$ and $\hat{U}$, are said $C$-isomorphic, for $C \subseteq U$, $\hat{U}$, written $D \sim_C D$, iff there exists a bijection $i : \text{adom}(D) \cup C \mapsto \text{adom}(\hat{D}) \cup C$ that is the identity on $C$, and such that for every $j = 1, \ldots, n$, and for every $\vec{u} \in \text{adom}(D)^{a_i}$, $D \models P_j(\vec{u}) \iff \hat{D} \models P_j(i(\vec{u}))$, where $i(\vec{u}) \models \langle i(u_1), \ldots, i(u_j) \rangle$.

In words: Instances obtained by uniformly renaming the elements not in $C$.

E.g., for $C = \{1\}$, $i(1) = 1$, $i(2) = a$, $i(3) = b$, $i(4) = c$. 

![Diagram of D and \hat{D} instances](image)
Data Abstraction

Isomorphic instances (cont.)

Isomorphic instances have a notable (well-known) property:

**Lemma**

If $D \sim_C \hat{D}$ then for every FOL $\varphi$ s.t. $\text{const}(\varphi) \subseteq C$, $D \models \varphi \iff \hat{D} \models \varphi$.

- The “coloured instance” satisfies $\varphi$ iff all the instances isomorphic to it do.
- The “coloured” instance stands for infinitely many isomorphic instances (isomorphism type):
  - same values iff same colours
- IDEA: No FO (sub-)formula from $S$ or $\varphi$ can distinguish two $C_{S,\varphi}$-isomorphic instances
- Observation: for given $b$, only finitely many isomorphism types
Crux of the Result

Theorem

If $D \sim_{C_{S,\varphi}} \hat{D}$, every concrete transition $\langle D, D' \rangle$ has an abstract counterpart $\langle \hat{D}, \hat{D}' \rangle$ s.t. $D' \sim_{C_{S,\varphi}} \hat{D}'$, and viceversa.

Execution: $\alpha(\vec{u}) = \langle \pi(\vec{v}), \psi(\vec{w}) \rangle$
Crux of the Result
If-Part (Intuition)

Need to prove that there exist $\hat{v}, \hat{w}, \hat{D}'$ s.t.
(i) $\hat{D} \models \pi(\hat{v})$, (ii) $\hat{D} \oplus \hat{D}' \models \psi(\hat{w})$, and (iii) $D' \sim_{C_S,\varphi} \hat{D}'$

- See $\hat{u}$ as a (1-tuple) relation
- We can prove that there exists $\hat{D}'$ and $\hat{u}$, and a $C_{S,\varphi}$-isomorphism between
  $$\{D, D', \hat{u}\} \text{ and } \{\hat{D}, \hat{D}', \hat{u}\}$$
- This is enough, as $\pi$ and $\varphi$ are invariant wrt $C_{S,\varphi}$-isomorphic instances
Crux of the Result

If-Part (Intuition) Cont.

$C_{S, \varphi}$-isomorphism between $\{D, D', \vec{u}\}$ and $\{\hat{D}, \hat{D}', \hat{u}\}$:

1. Obtain $\hat{u}$ by renaming the elements in $\vec{u}$ according to $i$, $k$, and preserving (in)equalities — $\hat{U}$ contains enough elements.
2. Obtain $\hat{D}'$ by renaming the elements in $D'$ according to $i$ and $j$. 
Data Abstraction

Bisimilar Kripke Structures

**Definition (C-bisimilar Kripke structures)**

Given $\mathcal{K} = \langle \Sigma, D_0, \tau \rangle$, $\hat{\mathcal{K}} = \langle \hat{\Sigma}, \hat{D}_0, \hat{\tau} \rangle$, and $C$, $\mathcal{K}$ and $\hat{\mathcal{K}}$ are $C$-bisimilar ($\mathcal{K} \approx_C \hat{\mathcal{K}}$) iff there exists a relation $R \subseteq \Sigma \times \hat{\Sigma}$, called $C$ (-preserving) bisimulation, s.t. $\langle D_0, \hat{D}_0 \rangle \in R$, and if $\langle D, \hat{D} \rangle \in R$ then:

- $D \sim_C \hat{D}$;
- for all $D'$ s.t. $\tau(D, D')$ there exists $\hat{D}'$ s.t. $\hat{\tau}(\hat{D}, \hat{D}')$ and $\langle D', \hat{D}' \rangle \in R$;
- for all $\hat{D}'$ s.t. $\hat{\tau}(\hat{D}, \hat{D}')$ there exists $D'$ s.t. $\tau(D, D')$ and $\langle D', \hat{D}' \rangle \in R$.

**Example**

![Example Diagram]
Data Abstraction
Bisimilar Kripke Structures (cont.)

**Lemma**

If $\mathcal{K} \approx_C \hat{\mathcal{K}}$, for every FO-CTL ($\mu$-calc) sentence $\varphi$ such that $\text{const}(\varphi) \subseteq C$, 

$$\mathcal{K} \models \varphi \iff \hat{\mathcal{K}} \models \varphi.$$ 

That is, $C$-bisimilar Kripke structures cannot be distinguished by FO-CTL formulas using only constants from $C$. Thus

- If $\hat{\mathcal{K}}$ is finite-state, we are able to check whether $\mathcal{K} \models \varphi$
- (In this case each $\hat{\mathcal{K}}$ transition abstracts infinitely many $\mathcal{K}$-transitions)
Lemma

Consider a $b$-bounded $S = \langle D, U, D_0, \Phi \rangle$, and a FO-CTL formula $\varphi$. Let:

- $\hat{S}_{b,\varphi} = \langle D, \hat{U}, D_0, \Phi \rangle$ be the $(b, \varphi)$-bounded abstract system of $S$
- $\mathcal{K}_b$ be the model of $S$
- $\hat{\mathcal{K}}_{b,\varphi}$ be the model of $\hat{S}_{b,\varphi}$

Then

$$\mathcal{K}_b \cong_{c_{S,\varphi}} \hat{\mathcal{K}}_{b,\varphi}$$

Proof by induction:

- base case: $D_0$ is $c_{S,\varphi}$-isomorphic wrt itself
- induction step: crux of the result shown above

Given a $b$-bounded $S$ and $\varphi$,

$$\mathcal{K}_b \models \varphi \iff \hat{\mathcal{K}}_{b,\varphi} \models \varphi$$
Application to the General Case

Preservation Theorem

- What if $S$ is unbounded? (Apart from undecidability)

Observation: for fixed $b$, the $(b, \varphi)$-bounded abstract system $S_{b,\varphi}$ corresponds to an (infinite) fragment of $S$

![Image: Diagram of abstract system]

Preservation theorem for the existential fragment FO-ECTL.

$$\varphi ::= \phi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \text{EX} \varphi \mid \text{E} \varphi \cup \varphi$$

**Theorem**

*Given $S$, $b \geq |D_0|$, and a FO-ECTL formula $\varphi$, if $\hat{K}_{b,\varphi} \models \varphi$ then $S \models \varphi$.***

Observe we can iterate on $b$
Application to Deployed Systems

What if $S$ is unbounded?

- Actual machines are memory-bounded
- Executed artifact systems cannot exceed the memory bound
- We can verify the artifact system up to a given bound

Technically requires an additional step, but conceptually same approach as for bounded systems
Conclusion

- Problem originating in the context of Business Processes
- Related to verification of database-driven systems (cf. ICDT 09)
- Contribution to scarcely investigated field (verification of processes in presence of data)
- Abstraction-based approach to bounded verification
  - Decidability
  - Actual technique, complete wrt bounded version
  - Practically relevant: any system runs on an actual, memory-bounded machine
- Partial solution to general case:
  - satisfied FO-ECTL properties preserved from abstract bounded to concrete unbounded system
- High complexity, but:
  - comparable to similar work (sometime good practical performance)
  - current technique non-optimal, space for improvements
    - e.g., CEGAR [CGL94] applied to the abstract system?
Future Directions

1. Quantification across modal operators (bounded case)
   - $AG \ EF \ \forall x \exists y. P(x, y)$ ✓
   - $AG \ \forall x \ EF \exists y. P(x, y)$? Ongoing
     - Decidability? We conjecture so! (FO-CTL with active-domain quantification)
     - Complexity? (at least) double exponential

2. Extension to MAS, in the context of Quantified Interpreted Systems [BL09, BLP11]
   - Agents capture the actors that execute the actions
   - Epistemic operators: $K$ (Ongoing), $C$, $D$

3. Transfer results to settings with similar (low-level) semantics:
   - E.g., Situation Calculus Ongoing.

4. Unbounded systems: what for formulas practically relevant?
Questions?
Bibliography

Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Riccardo De Masellis, and Paolo Felli.
Foundations of Relational Artifacts Verification.
To appear.

Francesco Belardinelli and Alessio Lomuscio.
Quantified Epistemic Logics for Reasoning About Knowledge in Multi-Agent Systems.

Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi.
A Computationally-Grounded Semantics for Artifact-Centric Systems and Abstraction Results.

Edmund M. Clarke, Orna Grumberg, and David E. Long.
Model Checking and Abstraction.

Edmund M. Clarke, Orna Grumberg, and Doron A. Peled.
Model Checking.

David Cohn and Rick Hull.

Alin Deutsch, Rick Hull, Fabio Patrizi, and Victor Vianu.
Automatic Verification of Data-centric Business Processes.

Alin Deutsch, Liying Sui, and Victor Vianu.
Specification and Verification of Data-Driven Web Applications.
Model Checking

In one slide

Problem: check whether a finite-state transition-system satisfies a temporal specification [CGP00]

Linear-time: the system defines (infinite-length) runs

E.g., LTL: $\square \diamond hold$, $\neg \square \diamond tail$

Branching-time: the system defines an (infinite-depth) tree

E.g., CTL: $AG(hold \rightarrow EX(head) \land EX(tail))$
Model Checking
(Well... two!)

Model Checking for *finite systems* is very well understood.
The main challenge is *efficiency*, not decidability.

- **CTL:**
  - Check whether the property holds over the generated tree
  - PTIME-complete

- **LTL:**
  - Check whether the property holds over the generated runs
  - PSPACE-complete

- **CTL***:
  - Mixes the above
  - PSPACE-complete