Bisimulation problems for classes of processes BPA and BPP

Martin Kot

Department of Computer Science, FEI
Technical University of Ostrava
17. listopadu 15, Ostrava-Poruba 708 33
Czech Republic

26 October 2005

Verification of systems

Question

Is the implementation of the system correct with respect to the specification?

Standard techniques

- Testing
- Simulation

Disadvantage

- The number of possible behaviors of the system can be huge or infinite
- The correctness is not guaranteed

Formal methods

Ensure correctness of all possible behaviors

- By hand
- Partially automated
- Fully automated - not in full generality

Approaches

- Theorem proving
- Model checking
- Equivalence checking

Theorem proving

- Construction of formal proofs of correctness
- Theorem provers assist the user and do some simple steps
- The user has to guide the program to do the crucial steps
- Requires a lot of knowledge, skill and practice from the user

Model checking and equivalence checking

- Fully automatic
- Cannot be used to arbitrary programs
- Properties of models without the expressive power of Turing machines

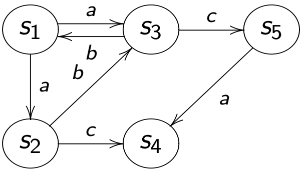
Model checking

Does the system satisfy a property expressed as a formula of some temporal logic?

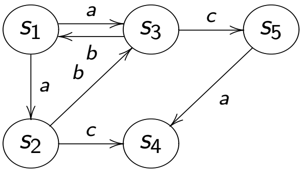
Equivalence checking

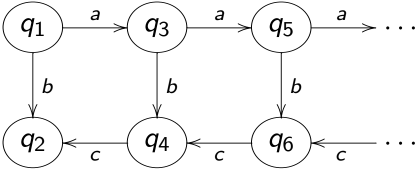
Are (descriptions of) two systems equivalent with respect to some equivalence?

Labeled transition system - examples



Labeled transition system - examples





Labeled transition system - definition

Definition (Labeled transition system)

Labeled transition system (LTS) is a tuple $(S, \mathcal{A}, \longrightarrow)$ where

- S is a (possibly infinite) set of states
- $\mathcal{A}$ is a set of actions
- $\longrightarrow \subseteq S \times \mathcal{A} \times S$ is a transition relation

Process rewrite systems

- Defined in the similar way as grammars
- Variables (nonterminals) may be connected using parallel or sequential composition
- Rules rewrite a term composed of variables into other term using an action
- Terms composed of variables using sequential composition have to be rewritten from the beginning of such sequence
- Terms composed of variables using parallel composition may be rewritten in arbitrary order
- Subclasses of PRS are defined using restrictions on the form of left-hand and right-hand sides of rules

Process rewrite systems - hierarchy

$$
\begin{tikzcd}[column sep=small]
 & & \begin{array}{c}(\mathcal{G},\mathcal{G})-PRS\\ PRS\end{array} \arrow[dl, dash] \arrow[dr, dash] & & \\
 & \begin{array}{c}(\mathcal{S},\mathcal{G})-PRS\\ PAD\end{array} \arrow[dl, dash] \arrow[dr, dash] & & \begin{array}{c}(\mathcal{P},\mathcal{G})-PRS\\ PAN\end{array} \arrow[dl, dash] \arrow[dr, dash] & \\
\begin{array}{c}(\mathcal{S},\mathcal{S})-PRS\\ PDA\end{array} \arrow[dr, dash] & & \begin{array}{c}(1,\mathcal{G})-PRS\\ PA\end{array} \arrow[dl, dash] \arrow[dr, dash] & & \begin{array}{c}(\mathcal{P},\mathcal{P})-PRS\\ PN\end{array} \arrow[dl, dash] \\
 & \begin{array}{c}(1,\mathcal{S})-PRS\\ BPA\end{array} \arrow[dr, dash] & & \begin{array}{c}(1,\mathcal{P})-PRS\\ BPP\end{array} \arrow[dl, dash] & \\
 & & \begin{array}{c}(1,1)-PRS\\ FS\end{array} & &
\end{tikzcd}
$$

Example of a BPA system

$$B \xrightarrow{b} BA$$
$$A \xrightarrow{a} \varepsilon$$
$$B \xrightarrow{b} \varepsilon$$

$$V = \{A, B\}$$
$$\mathcal{A} = \{a, b\}$$

Example of a BPA system

$$B \xrightarrow{b} BA \qquad A \xrightarrow{a} \varepsilon \qquad B \xrightarrow{b} \varepsilon$$

$$V = \{A, B\} \qquad \mathcal{A} = \{a, b\}$$

$$\begin{tikzcd}
B \arrow[r, "b"] \arrow[d, "b"] & BA \arrow[r, "b"] \arrow[d, "b"] & BAA \arrow[r, "b"] \arrow[d, "b"] & BAAA \arrow[r, "b"] \arrow[d, "b"] & \cdots \\
\varepsilon & A \arrow[l, "a"'] & AA \arrow[l, "a"'] & AAA \arrow[l, "a"'] & \cdots \arrow[l, "a"']
\end{tikzcd}$$

Basic Process Algebra (BPA)

Definition (BPA process)

A BPA process is given by a context-free grammar in Greibach normal form. Formally it is a triple $G = (V, \mathcal{A}, \Gamma)$, where

- V is a finite set variables (nonterminals),
- $\mathcal{A}$ is finite set of actions (terminals) and
- $\Gamma \subseteq V \times \mathcal{A} \times V^*$ is finite set of rewrite rules

Basic Process Algebra (BPA)

Definition (BPA process)

A BPA process is given by a context-free grammar in Greibach normal form. Formally it is a triple $G = (V, \mathcal{A}, \Gamma)$, where

- V is a finite set variables (nonterminals),
- $\mathcal{A}$ is finite set of actions (terminals) and
- $\Gamma \subseteq V \times \mathcal{A} \times V^*$ is finite set of rewrite rules

A BPA G gives rise to an LTS $\mathcal{S}_G = (V^*, \mathcal{A}, \longrightarrow)$ where:

- V^* is a sequence of variables of a BPA
- $\mathcal{A}$ the set of actions of a BPA
- $\longrightarrow$ is given by the rewrite rules extended with the prefix rewriting rule: if $X \xrightarrow{a} \alpha$ then $X\beta \xrightarrow{a} \alpha\beta$ for every $\beta \in V^*$.

Example of a BPP system

$$B \xrightarrow{b} B \| A$$
$$A \xrightarrow{a} \varepsilon$$
$$B \xrightarrow{b} \varepsilon$$

$$V = \{A, B\}$$
$$\mathcal{A} = \{a, b\}$$

Example of a BPP system

$$
\begin{aligned}
B &\xrightarrow{b} B\|A \\
A &\xrightarrow{a} \varepsilon \\
B &\xrightarrow{b} \varepsilon
\end{aligned}
\qquad\qquad
\begin{aligned}
V &= \{A, B\} \\
\mathcal{A} &= \{a, b\}
\end{aligned}
$$

$$
\begin{tikzcd}
B \arrow[r, shift left, "b"] \arrow[d, "b"] & A\|B \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & A\|A\|B \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & A\|A\|A\|B \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & \cdots \arrow[l, shift left, "a"] \\
\varepsilon & A \arrow[l, "a"] & A\|A \arrow[l, "a"] & A\|A\|A \arrow[l, "a"] & \cdots \arrow[l, "a"]
\end{tikzcd}
$$

Basic Parallel Processes (BPP)

Definition (BPP process)

A BPP process is given by a context-free grammar in Greibach normal form. Formally it is a triple $G = (V, \mathcal{A}, \Gamma)$, where

- V is a finite set variables (nonterminals),
- $\mathcal{A}$ is finite set of actions (terminals) and
- $\Gamma \subseteq V \times \mathcal{A} \times V^*$ is finite set of rewrite rules

Basic Parallel Processes (BPP)

Definition (BPP process)

A BPP process is given by a context-free grammar in Greibach normal form. Formally it is a triple $G = (V, \mathcal{A}, \Gamma)$, where

- V is a finite set variables (nonterminals),
- $\mathcal{A}$ is finite set of actions (terminals) and
- $\Gamma \subseteq V \times \mathcal{A} \times V^*$ is finite set of rewrite rules

A BPP G gives rise to an LTS $\mathcal{S}_G = (V^*, \mathcal{A}, \longrightarrow)$ where:

- V^* is a multiset of variables of a BPP
- $\mathcal{A}$ the set of actions of a BPP
- $\longrightarrow$ is given by the rewrite rules extended with the rule: if $X \xrightarrow{a} \alpha$ then $\beta_1 \| X \| \beta_2 \xrightarrow{a} \beta_1 \| \alpha \| \beta_2$ for every $\beta_1, \beta_2 \in V^*$.

Bisimulation equivalence

Definition (Bisimulation)

Given an LTS $(S, \mathcal{A}, \longrightarrow)$, a binary relation $\mathcal{R} \subseteq S \times S$ is a bisimulation iff for each $(s, t) \in \mathcal{R}$ and $a \in \mathcal{A}$ we have:

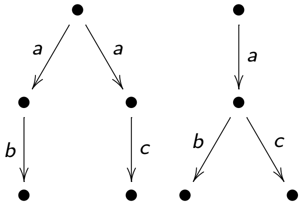
- $\forall s' \in S : s \xrightarrow{a} s' \Rightarrow (\exists t' : t \xrightarrow{a} t' \wedge (s', t') \in \mathcal{R})$, and
- $\forall t' \in S : t \xrightarrow{a} t' \Rightarrow (\exists s' : s \xrightarrow{a} s' \wedge (s', t') \in \mathcal{R})$.

Bisimulation equivalence

Definition (Bisimulation)

Given an LTS $(S, \mathcal{A}, \longrightarrow)$, a binary relation $\mathcal{R} \subseteq S \times S$ is a bisimulation iff for each $(s, t) \in \mathcal{R}$ and $a \in \mathcal{A}$ we have:

- $\forall s' \in S : s \xrightarrow{a} s' \Rightarrow (\exists t' : t \xrightarrow{a} t' \wedge (s', t') \in \mathcal{R})$, and
- $\forall t' \in S : t \xrightarrow{a} t' \Rightarrow (\exists s' : s \xrightarrow{a} s' \wedge (s', t') \in \mathcal{R})$.

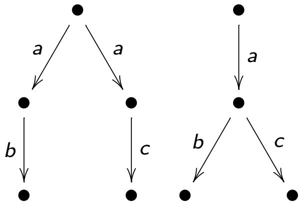


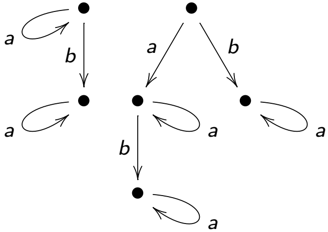
Bisimulation equivalence

Definition (Bisimulation)

Given an LTS $(S, \mathcal{A}, \longrightarrow)$, a binary relation $\mathcal{R} \subseteq S \times S$ is a **bisimulation** iff for each $(s, t) \in \mathcal{R}$ and $a \in \mathcal{A}$ we have:

- $\forall s' \in S : s \xrightarrow{a} s' \Rightarrow (\exists t' : t \xrightarrow{a} t' \wedge (s', t') \in \mathcal{R})$, and
- $\forall t' \in S : t \xrightarrow{a} t' \Rightarrow (\exists s' : s \xrightarrow{a} s' \wedge (s', t') \in \mathcal{R})$.





Bisimulation equivalence

Definition (Bisimulation)

Given an LTS $(S, \mathcal{A}, \longrightarrow)$, a binary relation $\mathcal{R} \subseteq S \times S$ is a bisimulation iff for each $(s, t) \in \mathcal{R}$ and $a \in \mathcal{A}$ we have:

- $\forall s' \in S : s \xrightarrow{a} s' \Rightarrow (\exists t' : t \xrightarrow{a} t' \wedge (s', t') \in \mathcal{R})$, and
- $\forall t' \in S : t \xrightarrow{a} t' \Rightarrow (\exists s' : s \xrightarrow{a} s' \wedge (s', t') \in \mathcal{R})$.

Definition (Bisimulation equivalence)

The bisimulation equivalence (bisimilarity) $\sim$ is the maximal bisimulation (i.e. union of all bisimulations).

Bisimularity of two different systems Σ_1 and Σ_2 is defined as bisimilarity of their initial states on disjoint union of Σ_1 and Σ_2.

Bisimilarity problems

Bisimilarity of two processes

Instance: Definitions of two processes (possibly from different classes) with initial variables X and Y

Question: Is $X \sim Y$?

Existence of a bisimilar process from the different class

Instance: Definition of a process from one class with initial variable X

Question: Does a process from the given other class exist such that for its initial variable Y holds $X \sim Y$?

Bisimilarity problems for BPP and BPA

- Bisimilarity of BPP is PSPACE-complete - *Srba; Jančar*
- Regularity (existence of bisimilar finite-state system) of a BPP is PSPACE-complete - *Srba; Kot*
- Bisimilarity of a BPP with finite-state process is in $O(n^4)$ - *Kot, Sawa*

- Bisimilarity of BPA is PSPACE-hard and in 2-EXPTIME- *Srba; Burkart, Caucal, Steffen*
- Regularity of a BPA is PSPACE-hard and in 2-EXPTIME- *Srba; Burkart, Caucal, Steffen*
- Bisimilarity of a BPA with finite-state process is P-complete - *Balcazar, Gabarro, Santha; Kučera, Mayr*

Bisimilarity problems between BPP and BPA - known results

- Existence of a bisimilar nBPA to a given nBPP is decidable in polynomial time - *Černá, Křetínský, Kučera*
- Existence of a bisimilar nBPP to a given nBPA is decidable in polynomial time - *Černá, Křetínský, Kučera*
- Bisimilarity of a nBPP with a nBPA is decidable - *Černá, Křetínský, Kučera*
- Bisimilarity of a BPP with a BPA is decidable - *Moller, Jančar, Kučera*

Bisimilarity problems between BPP and BPA - our results

Bisimilarity of BPA and BPP

- In 3-EXPTIME

Given a BPA system, does a bisimilar BPP exist

- PSPACE-hard - reduction from regularity problem for BPA
- Semi-decidable

Given a BPP system, does a bisimilar BPA exist

- PSPACE-hard - reduction from regularity problem for BPP
- PSPACE-complete - algorithm which checks a condition on the BPP in polynomial space. A bisimilar BPA exist iff condition is satisfied

Bisimilarity between BPP and BPA

Algorithm

- Check, if there exists some BPA system bisimilar with given BPP
- If BPA exist, construct BPP in the normal form of an exponential size to the given BPP
- Construct a BPA bisimilar with BPP in the normal form of the same asymptotic size
- Check bisimilarity of constructed BPA and given BPA

Publication

Jančar, Kot, Sawa – Notes on Complexity of Bisimilarity between BPA and BPP, EXPRESS'05 - Affiliated workshop of CONCUR'05, San Francisco

Bisimilar BPP and BPA

$$
\begin{aligned}
B &\xrightarrow{b} B\|A \\
A &\xrightarrow{a} \varepsilon \\
B &\xrightarrow{b} C \\
C &\xrightarrow{a} C
\end{aligned}
$$

$$
\begin{tikzcd}
B \arrow[r, shift left, "b"] \arrow[d, "b"] & A\|B \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & A\|A\|B \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & A\|A\|A\|B \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & \cdots \arrow[l, shift left, "a"] \\
C \arrow[loop below, "a"] & A\|C \arrow[l, "a"] \arrow[loop below, "a"] & A\|A\|C \arrow[l, "a"] \arrow[loop below, "a"] & A\|A\|A\|C \arrow[l, "a"] \arrow[loop below, "a"] & \cdots \arrow[l, "a"]
\end{tikzcd}
$$

Bisimilar BPP and BPA

$$
\begin{aligned}
D &\xrightarrow{b} ED\\
E &\xrightarrow{b} EE\\
E &\xrightarrow{a} \varepsilon\\
D &\xrightarrow{b} F\\
E &\xrightarrow{b} F\\
F &\xrightarrow{a} F
\end{aligned}
$$

$$
\begin{tikzcd}
D \arrow[r, shift left, "b"] \arrow[d, "b"] & ED \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & EED \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & EEED \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & \cdots \arrow[l, shift left, "a"] \\
F \arrow[loop below, "a"] & FD \arrow[loop below, "a"] & FED \arrow[loop below, "a"] & FEED \arrow[loop below, "a"] &
\end{tikzcd}
$$

Future work

- Find a necessary and sufficient condition for a BPA to be bisimilar with some BPP and hence show decidability (and possibly complexity) of deciding existence of a BPP bisimilar to given BPA
- Improve complexity bound for checking bisimilarity between BPA and BPP
- Extend these results to bisimilarity between PDA and BPP

Thank you

A BPP system with independently growing functions

$$A \xrightarrow{a} AA$$
$$A \xrightarrow{a} BA$$
$$A \xrightarrow{b} \varepsilon$$
$$B \xrightarrow{c} \varepsilon$$

$$\begin{tikzcd}
\varepsilon & A \arrow[l, "b"] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & AA \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & AAA \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & \cdots \arrow[l, shift left, "b"] \\
B \arrow[u, "c"'] & BA \arrow[l, "b"] \arrow[u, shift right, "c"'] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & BAA \arrow[l, shift left, "b"] \arrow[u, shift right, "c"'] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & BAAA \arrow[l, shift left, "b"] \arrow[u, shift right, "c"'] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & \cdots \arrow[l, shift left, "b"] \\
BB \arrow[u, "c"'] & BBA \arrow[l, "b"] \arrow[u, shift right, "c"'] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & BBAA \arrow[l, shift left, "b"] \arrow[u, shift right, "c"'] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & BBAAA \arrow[l, shift left, "b"] \arrow[u, shift right, "c"'] \arrow[r, shift left, "a"] \arrow[d, shift right, "a"'] & \cdots \arrow[l, shift left, "b"] \\
\vdots \arrow[u, "c"'] & \vdots \arrow[u, shift right, "c"'] & \vdots \arrow[u, shift right, "c"'] & \vdots \arrow[u, shift right, "c"'] &
\end{tikzcd}$$

A BPP system with one function overgrowing the other

$$B \xrightarrow{b} BA$$
$$A \xrightarrow{a} \varepsilon$$
$$B \xrightarrow{b} \varepsilon$$

$$\begin{tikzcd}
B \arrow[r, shift left, "b"] \arrow[d, "b"] & AB \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & AAB \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & AAAB \arrow[l, shift left, "a"] \arrow[r, shift left, "b"] \arrow[d, "b"] & \cdots \arrow[l, shift left, "a"] \\
\varepsilon & A \arrow[l, "a"] & AA \arrow[l, "a"] & AAA \arrow[l, "a"] & \cdots \arrow[l, "a"]
\end{tikzcd}$$