On Decidability of the Bisimilarity on Higher-order Processes with Parameterization

Xu Xian (徐贤)

(East China University of Science and Technology)

&

Zhang Wenbo (张文博)

(Shanghai Ocean University)

EXPRESS/SOS 2021

August 23rd, 2021

What is this work about?

The decidability of parameterized higher-order processes.

What is this work about?

The decidability of parameterized higher-order processes.

higher-order	=	process-passing (no name-passing)
parameterized	=	a way to promote expressiveness [Lanese et al., 2010]

What is parameterization?

It is composed of abstraction and application (pursuant to those in lambda-calculus).

What is parameterization?

It is composed of abstraction and application (pursuant to those in lambda-calculus).

Two kinds of parameterization extending Π^{m}

- Π^{m}: $0 \mid X \mid u(X).P \mid \overline{u}P'.P \mid P \,|\, P'$

What is parameterization?

It is composed of abstraction and application (pursuant to those in lambda-calculus).

Two kinds of parameterization extending Π^{m}

- Π^{m}: $0 \mid X \mid u(X).P \mid \overline{u}P'.P \mid P \,|\, P'$
- Parameterization on processes:

  abstraction: $\langle X \rangle P$ (X is a process variable) ($\lambda X.P$)

  application: $P\langle Q \rangle$

What is parameterization?

It is composed of abstraction and application (pursuant to those in lambda-calculus).

Two kinds of parameterization extending Π^{m}

- Π^{m}: $0 \mid X \mid u(X).P \mid \overline{u}P'.P \mid P \,|\, P'$
- Parameterization on processes:

  abstraction: $\langle X \rangle P$ (X is a process variable) ($\lambda X.P$)

  application: $P\langle Q \rangle$
- Parameterization on names:

  abstraction: $\langle x \rangle P$ (x is a name variable)

  application: $P\langle u \rangle$

What is parameterization?

It is composed of abstraction and application (pursuant to those in lambda-calculus).

Two kinds of parameterization extending Π^{m}

- Π^{m}: $\quad 0 \;\big|\; X \;\big|\; u(X).P \;\big|\; \overline{u}P'.P \;\big|\; P \,|\, P'$
- Parameterization on processes:

  abstraction: $\langle X\rangle P$ $\qquad$ (X is a process variable) $\qquad$ ($\lambda X.P$)

  application: $P\langle Q\rangle$
- Parameterization on names:

  abstraction: $\langle x\rangle P$ $\qquad$ (x is a name variable)

  application: $P\langle u\rangle$
- Π^{mp}: Π extended with both kinds of parameterization

Main results

- In Π^{mp}, the strong bisimilarity is decidable.
- For the strong bisimilarity, an axiomatization and a bisimilarity checking algorithm.

Organization

- DECIDABILITY OF THE STRONG BISIMILARITY IN Π^{mp}
- AXIOMATIZATION
- BISIMILARITY CHECKING ALGORITHM

1 Decidability of the strong bisimilarity in Π^{mp}

The strong bisimilarities

1. Strong HO-IO bisimilarity $\sim^{\circ}_{\mathrm{hoio}}$.

2. Strong HO bisimilarity $\sim_{\mathrm{ho}}$.

3. Strong context bisimilarity $\sim_{\mathrm{ctx}}$.

4. Strong normal bisimilarity $\sim_{\mathrm{nr}}$.

5. Open strong normal bisimilarity $\sim^{\circ}_{\mathrm{nr}}$.

Main results

1. $\sim^{\circ}_{\text{hoio}}$ is decidable.

2. All the strong bisimilarities coincide.

$$\begin{tikzcd}
\sim^{\circ}_{\text{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\text{ho}} \arrow[r, dotted, "(2)"] & \sim_{\text{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\text{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\text{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}$$

Main results

1. $\sim^{\circ}_{\text{hoio}}$ is decidable.

2. All the strong bisimilarities coincide.

$$
\begin{tikzcd}
\sim^{\circ}_{\text{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\text{ho}} \arrow[r, dotted, "(2)"] & \sim_{\text{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\text{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\text{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}
$$

- Similar results for Π^{m} are first given by Lanese et al. [Lanese et al, 2011].

  In this work, we adapt the approach of that work to accommodate parameterization.

Definition 1 (Strong HO-IO bisimilarity). A symmetric binary relation $\mathcal{R}$ over Π^{mp} terms is a strong HO-IO bisimulation, if whenever $P\,\mathcal{R}\,Q$ the following properties hold.

1. If P is a non-abstraction, then so is Q.
2. If P is a process-abstraction $\langle Y\rangle A$, then Q is a process-abstraction $\langle Y\rangle B$, and $A\,\mathcal{R}\,B$.
3. If P is a name-abstraction $\langle y\rangle A$, then Q is a name-abstraction $\langle y\rangle B$, and $A\,\mathcal{R}\,B$.
4. If $P\xrightarrow{\overline{a}A}P'$, then $Q\xrightarrow{\overline{a}B}Q'$ with $A\,\mathcal{R}\,B$ and $P'\,\mathcal{R}\,Q'$.
5. If $P\xrightarrow{a(X)}P'$, then $Q\xrightarrow{a(X)}Q'$ and $P'\,\mathcal{R}\,Q'$.
6. If $P\equiv X\,|\,P'$, then $Q\equiv X\,|\,Q'$ and $P'\,\mathcal{R}\,Q'$.
7. If $P\equiv X\langle A\rangle\,|\,P'$, then $Q\equiv X\langle B\rangle\,|\,Q'$ with $A\,\mathcal{R}\,B$ and $P'\,\mathcal{R}\,Q'$.
8. If $P\equiv X\langle d\rangle\,|\,P'$, then $Q\equiv X\langle d\rangle\,|\,Q'$ and $P'\,\mathcal{R}\,Q'$.

The strong HO-IO bisimilarity, $\sim^{\circ}_{\mathrm{hoio}}$, is the largest strong HO-IO bisimulation.

Definition 2 (Depth of a term)**.** We define depth(P) as follows.

P	$\mathrm{depth}(P)$	
0	0	
X	1	
$m(X).P_1$	$\mathrm{depth}(P_1)+1$	
$\overline{m}(P_1)$	$\mathrm{depth}(P_1)+1$	
$P_1 \mid P_2$	$\mathrm{depth}(P_1)+\mathrm{depth}(P_2)$	
$\langle X\rangle P_1$	$\mathrm{depth}(P_1)+1$	
$X\langle P_1\rangle$	$\mathrm{depth}(P_1)+1$	
$P_1\langle P_2\rangle$	$\mathrm{depth}(P_3\{P_2/Y\})$	where P_1 is $\langle Y\rangle P_3$
$\langle x\rangle P_1$	$\mathrm{depth}(P_1)+1$	
$X\langle n\rangle$	1	
$P_1\langle n\rangle$	$\mathrm{depth}(P_3\{n/y\})$	where P_1 is $\langle y\rangle P_3$

Lemma 3. $\sim^{\circ}_{hoio}$ *is decidable.*

Proof (sketch). We decide whether $P \sim^{\circ}_{\text{hoio}} Q$ by induction on $\text{depth}(P)$.

1. If P *is an abstraction*, then check Q is also an abstraction accordingly, and continue by induction hypothesis.

2. If P *does an action*, then check Q can simulate, and continue by induction hypothesis.

3. If P *reveals some free variable*, say X, then check Q also does with X, and continue by induction hypothesis.

□

Coincidence between the strong bisimilarities

$$
\begin{tikzcd}
\sim^{\circ}_{\mathrm{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\mathrm{ho}} \arrow[r, dotted, "(2)"] & \sim_{\mathrm{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\mathrm{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\mathrm{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}
$$

Proof (sketch).

1. Main difference between $\sim^{\circ}_{\mathrm{hoio}}$ and $\sim_{\mathrm{ho}}$: $\sim_{\mathrm{ho}}$ requires closed under substitution and τ simulation.

2. Approach: show that the following relation is a strong HO bisimulation.

$$\mathcal{R}_1 \stackrel{\mathrm{def}}{=} \{(P,Q) \mid P \sim^{\circ}_{\mathrm{hoio}} Q\} \cup \sim^{\circ}_{\mathrm{hoio}}$$

3. Technical lemmas:

   (a) If $P \sim^{\circ}_{\mathrm{hoio}} Q$, then $P\{R/X\} \sim^{\circ}_{\mathrm{hoio}} Q\{R/X\}$.

   (b) If $P \sim^{\circ}_{\mathrm{hoio}} Q$ and $P \xrightarrow{\tau} P'$, then $Q \xrightarrow{\tau} Q'$ and $P' \sim^{\circ}_{\mathrm{hoio}} Q'$.

$\square$

Coincidence between the strong bisimilarities

$$\begin{tikzcd}
\sim^{\circ}_{\text{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\text{ho}} \arrow[r, dotted, "(2)"] & \sim_{\text{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\text{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\text{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}$$

Proof (sketch).

1. Main difference between $\sim_{\text{ho}}$ and $\sim_{\text{ctx}}$: the output simulation.

$\sim_{\text{ho}}$: If $P \xrightarrow{\overline{a}A} P'$, then $Q \xrightarrow{\overline{a}B} Q'$ with $A \sim_{\text{ho}} B$ and $P' \sim_{\text{ho}} Q'$.

$\sim_{\text{ctx}}$: If $P \xrightarrow{\overline{a}A} P'$, then $Q \xrightarrow{\overline{a}B} Q'$ and for every context E, it holds that $E(A) \mid P' \sim_{\text{ctx}} E(B) \mid Q'$.

2. Intuition: $\sim_{\text{ctx}}$ requires closure under contexts.

3. Approach: use the congruence property of $\sim_{\text{ho}}$.

$\square$

Coincidence between the strong bisimilarities

$$
\begin{tikzcd}
\sim^{\circ}_{\text{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\text{ho}} \arrow[r, dotted, "(2)"] & \sim_{\text{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\text{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\text{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}
$$

Proof (sketch).

The triggers family [Sangiorgi, 1992; Xu, 2013 & 2020]:

$$
\boxed{① \ \mathrm{Tr}_m \overset{\text{def}}{=} \overline{m}, \qquad ② \ \mathrm{Tr}^D_m \overset{\text{def}}{=} \langle Z\rangle\overline{m}Z, \qquad ③ \ \mathrm{Tr}^{D,d}_m \overset{\text{def}}{=} \langle z\rangle\overline{m}[\langle Z\rangle(Z\langle z\rangle)]}
$$

$\square$

Coincidence between the strong bisimilarities

$$\begin{tikzcd} \sim^{\circ}_{\mathrm{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\mathrm{ho}} \arrow[r, dotted, "(2)"] & \sim_{\mathrm{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\mathrm{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\mathrm{nr}} \arrow[llll, dotted, bend left, "(5)"] \end{tikzcd}$$

Proof (sketch).

The triggers family [Sangiorgi, 1992; Xu, 2013 & 2020]:

$$\boxed{① \ \mathrm{Tr}_m \stackrel{\mathrm{def}}{=} \overline{m}, \qquad ② \ \mathrm{Tr}^D_m \stackrel{\mathrm{def}}{=} \langle Z \rangle \overline{m} Z, \qquad ③ \ \mathrm{Tr}^{D,d}_m \stackrel{\mathrm{def}}{=} \langle z \rangle \overline{m}[\langle Z \rangle (Z \langle z \rangle)]}$$

1. Main difference between $\sim_{\mathrm{ctx}}$ and $\sim_{\mathrm{nr}}$: closure under substitution and output simulation.

$\sim_{\mathrm{ctx}}$: requires closure for *any processes* for input, and *any contexts* for output.

$\sim_{\mathrm{nr}}$: requires closure for *triggers* for input, and *trigger-receiving contexts* for output.

2. Intuition: $\sim_{\mathrm{nr}}$ is a special case of $\sim_{\mathrm{ctx}}$.

3. Approach: in $\sim_{\mathrm{ctx}}$, choose special forms of processes or contexts based on triggers.

$\square$

Coincidence between the strong bisimilarities

$$\begin{tikzcd}[column sep=huge] \sim^{\circ}_{\mathrm{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\mathrm{ho}} \arrow[r, dotted, "(2)"] & \sim_{\mathrm{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\mathrm{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\mathrm{nr}} \arrow[llll, dotted, bend left, "(5)"] \end{tikzcd}$$

Proof (sketch).

1. Main difference between $\sim_{\mathrm{nr}}$ and $\sim^{\circ}_{\mathrm{nr}}$: closure under substitution and simulation of open terms over free variables.

$\sim_{\mathrm{nr}}$: requires closure under substitution of triggers, while $\sim^{\circ}_{\mathrm{nr}}$ does not.

$\sim^{\circ}_{\mathrm{nr}}$: requires direct matching of free variables, while $\sim_{\mathrm{nr}}$ does not.

□

Coincidence between the strong bisimilarities

$$\begin{tikzcd}
\sim^{\circ}_{\text{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\text{ho}} \arrow[r, dotted, "(2)"] & \sim_{\text{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\text{nr}} \arrow[r, dotted, "\color{red}(4)"] & \sim^{\circ}_{\text{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}$$

Proof (sketch).

1. Main difference between $\sim_{\text{nr}}$ and $\sim^{\circ}_{\text{nr}}$: closure under substitution and simulation of open terms over free variables.

   $\sim_{\text{nr}}$: requires closure under substitution of triggers, while $\sim^{\circ}_{\text{nr}}$ does not.

   $\sim^{\circ}_{\text{nr}}$: requires direct matching of free variables, while $\sim_{\text{nr}}$ does not.

2. Intuition: use the closure property of $\sim^{\circ}_{\text{nr}}$ to build a bisimulation.

3. Technical property: $\sim^{\circ}_{\text{nr}}$ is closed under substitution (from its definition). That is,

$$P \sim_{\text{nr}} Q$$

if and only if

$$P\{\widetilde{\mathrm{Tr}_{m_1}}/\widetilde{X_1}\}\{\widetilde{\mathrm{Tr}^{D}_{m_2}}/\widetilde{X_2}\}\{\widetilde{\mathrm{Tr}^{D,d}_{m_3}}/\widetilde{X_3}\} \sim_{\text{nr}} Q\{\widetilde{\mathrm{Tr}_{m_1}}/\widetilde{X_1}\}\{\widetilde{\mathrm{Tr}^{D}_{m_2}}/\widetilde{X_2}\}\{\widetilde{\mathrm{Tr}^{D,d}_{m_3}}/\widetilde{X_3}\}$$

$\square$

Coincidence between the strong bisimilarities

$$
\begin{tikzcd}
\sim^{\circ}_{\mathrm{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\mathrm{ho}} \arrow[r, dotted, "(2)"] & \sim_{\mathrm{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\mathrm{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\mathrm{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}
$$

Proof (sketch).

1. Main difference between $\sim^{\circ}_{\mathrm{nr}}$ and $\sim^{\circ}_{\mathrm{hoio}}$: the output simulation and the simulation of an open process with a free variable for process-abstraction. We take a case (the others are similar).

$\sim^{\circ}_{\mathrm{nr}}$: If $P \xrightarrow{\overline{a}A} P'$ in which A is a process abstraction, then $Q \xrightarrow{\overline{a}B} Q'$ for process-abstraction B, and it holds for fresh m that

$$m(Z).A\langle Z\rangle \mid P' \ \sim^{\circ}_{\mathrm{nr}} \ m(Z).B\langle Z\rangle \mid Q' \tag{$\ast$a}$$

$\sim^{\circ}_{\mathrm{hoio}}$: If $P \xrightarrow{\overline{a}A} P'$, then $Q \xrightarrow{\overline{a}B} Q'$ with

$$A \sim^{\circ}_{\mathrm{hoio}} B \quad \text{and} \quad P' \sim^{\circ}_{\mathrm{hoio}} Q' \tag{$\ast$b}$$

$\square$

Coincidence between the strong bisimilarities

$$\begin{tikzcd}
\sim^{\circ}_{\text{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\text{ho}} \arrow[r, dotted, "(2)"] & \sim_{\text{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\text{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\text{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}$$

Proof (sketch).

1. Main difference between $\sim^{\circ}_{\text{nr}}$ and $\sim^{\circ}_{\text{hoio}}$: the output simulation and the simulation of an open process with a free variable for process-abstraction. We take a case (the others are similar).

$\sim^{\circ}_{\text{nr}}$: If $P \xrightarrow{\overline{a}A} P'$ in which A is a process abstraction, then $Q \xrightarrow{\overline{a}B} Q'$ for process-abstraction B, and it holds for fresh m that

$$m(Z).A\langle Z\rangle \mid P' \;\sim^{\circ}_{\text{nr}}\; m(Z).B\langle Z\rangle \mid Q' \qquad (\ast\text{a})$$

$\sim^{\circ}_{\text{hoio}}$: If $P \xrightarrow{\overline{a}A} P'$, then $Q \xrightarrow{\overline{a}B} Q'$ with

$$A \sim^{\circ}_{\text{hoio}} B \quad \text{and} \quad P' \sim^{\circ}_{\text{hoio}} Q' \qquad (\ast\text{b})$$

2. Intuition: use the fresh name to separate the parallel components, so as to extract the two pairs of bisimilar processes.

$\square$

Coincidence between the strong bisimilarities

$$\begin{tikzcd}
\sim^{\circ}_{\mathrm{hoio}} \arrow[r, dotted, "(1)"] & \sim_{\mathrm{ho}} \arrow[r, dotted, "(2)"] & \sim_{\mathrm{ctx}} \arrow[r, dotted, "(3)"] & \sim_{\mathrm{nr}} \arrow[r, dotted, "(4)"] & \sim^{\circ}_{\mathrm{nr}} \arrow[llll, dotted, bend left, "(5)"]
\end{tikzcd}$$

Proof (sketch).

1. Main difference between $\sim^{\circ}_{\mathrm{nr}}$ and $\sim^{\circ}_{\mathrm{hoio}}$: the output simulation and the simulation of an open process with a free variable for process-abstraction. We take a case (the others are similar).

$\sim^{\circ}_{\mathrm{nr}}$: If $P \xrightarrow{\overline{a}A} P'$ in which A is a process abstraction, then $Q \xrightarrow{\overline{a}B} Q'$ for process-abstraction B, and it holds for fresh m that

$$m(Z).A\langle Z\rangle \,|\, P' \sim^{\circ}_{\mathrm{nr}} m(Z).B\langle Z\rangle \,|\, Q' \qquad (\ast\mathrm{a})$$

$\sim^{\circ}_{\mathrm{hoio}}$: If $P \xrightarrow{\overline{a}A} P'$, then $Q \xrightarrow{\overline{a}B} Q'$ with

$$A \sim^{\circ}_{\mathrm{hoio}} B \quad \text{and} \quad P' \sim^{\circ}_{\mathrm{hoio}} Q' \qquad (\ast\mathrm{b})$$

2. Intuition: use the fresh name to separate the parallel components, so as to extract the two pairs of bisimilar processes.

3. Technical lemma: Assume that m is fresh.

If $m(Z).A\langle Z\rangle \,|\, P' \sim^{\circ}_{\mathrm{nr}} m(Z).B\langle Z\rangle \,|\, Q'$, then $A \sim^{\circ}_{\mathrm{nr}} B$ and $P' \sim^{\circ}_{\mathrm{nr}} Q'$.

$\square$

The strong bisimilarities

$$\begin{tikzcd}
\sim^{\circ}_{\mathrm{hoio}} \arrow[r, dotted] & \sim_{\mathrm{ho}} \arrow[r, dotted] & \sim_{\mathrm{ctx}} \arrow[r, dotted] & \sim_{\mathrm{nr}} \arrow[r, dotted] & \sim^{\circ}_{\mathrm{nr}} \arrow[llll, dotted, bend left]
\end{tikzcd}$$

Theorem 4. *All the strong bisimilarities above are coincident and decidable.*

2 Axiomatization

The axiom system

The axiom system is composed of the following laws.

1. Structural congruence.

$$(P \mid Q) \mid R = P \mid (Q \mid R) \qquad P \mid Q = Q \mid P \qquad P \mid 0 = P$$

2. Distribution [Lanese et al., 2011].

$$a(X).\left(P \mid \prod^{k-1} a(X).P\right) = \prod^{k} a(X).P$$

3. Parameterization.

$$(\langle X \rangle P)\langle Q \rangle = P\{Q/X\} \qquad (\langle x \rangle P)\langle m \rangle = P\{m/x\}$$

The axiom system

The axiom system is composed of the following laws.

1. Structural congruence.

$$(P \mid Q) \mid R = P \mid (Q \mid R) \qquad P \mid Q = Q \mid P \qquad P \mid 0 = P$$

2. Distribution [Lanese et al., 2011].

$$a(X).\left(P \mid \prod^{k-1} a(X).P\right) = \prod^{k} a(X).P$$

3. Parameterization.

$$(\langle X \rangle P)\langle Q \rangle = P\{Q/X\} \qquad (\langle x \rangle P)\langle m \rangle = P\{m/x\}$$

Normal form P is in *normal form* if it cannot be rewritten by the laws for Distribution and Parameterization (left to right). Any P has a *unique normal form up to* $\equiv$, denoted as $\mathrm{nf}(P)$.

Completeness of the axiom system

Lemma 5 (Completeness). *For any P, Q, if $P \sim Q$ then $P = Q$, more precisely,* $\mathrm{nf}(P) = \mathrm{nf}(Q)$.

Proof (sketch). Induction on $\mathrm{depth}(P)$.

1. Key lemmas:
   (a) About normal forms: $P \sim \mathrm{nf}(P)$.
   (b) About input prefix:
   If $a(X).P \sim Q \mid Q' \quad (Q, Q' \nsim 0)$, then $a(X).P \sim \prod_{i=1}^{k} a(X).A$ ($k > 1$) with $a(X).A$ is in nf.

2. A notion of *prime* processes [Lanese et al., 2011; Milner et al., 1993].
   (a) P is *prime* if $P \nsim 0$ and $P \sim P_1 \mid P_2$ implies $P_1 \sim 0$ or $P_2 \sim 0$.
   (b) If $P \sim \prod_{i=1}^{n} P_i$ where each P_i is prime, $\prod_{i=1}^{n} P_i$ is called a *prime decomposition* of P.

3. Properties about prime processes.
   (a) If P is a prefixed process in normal form, then P is prime.
   (b) Unique prime decomposition.
   If P has two prime decompositions $P \sim \prod_{i=1}^{n} P_i$ and $P \sim \prod_{j=1}^{m} Q_j$, then $n = m$ and there is a permutation $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, such that $P_i \sim Q_{\sigma(i)}$ for each $i \in \{1, 2, \ldots, n\}$.

□

3 Bisimilarity checking algorithm

Core of the algorithm

Input Two processes P and Q.

Procedure

1. (Data structure) Representing processes P and Q as trees.

2. (Transformation) Normalizing the trees.

3. (Comparison) Comparing the trees up-to syntax.

Output Whether $P = Q$ or not.

Core of the algorithm

$\boxed{\text{Input}}$ Two processes P and Q.

Procedure

1. (Data structure) Representing processes P and Q as trees.
2. (Transformation) Normalizing the trees.
   (a) Rewrite each process by the application rules, if possible.
   (b) Normalize each process over parallel composition.
   (c) Apply the distribution law, if possible.
3. (Comparison) Comparing the trees up-to syntax.

$\boxed{\text{Output}}$ Whether $P = Q$ or not.

Tree representation

Definition 6 (Tree representation)**.** The tree representation of P is defined as follows.

- $\mathsf{Tree}(0) = 0[\,]$
- $\mathsf{Tree}(X) = \mathsf{db}(X)[\,]$
- $\mathsf{Tree}(a(X).P) = a[\mathsf{Tree}(P)]$
- $\mathsf{Tree}(\overline{a}(Q)) = a^{O}[\mathsf{Tree}(Q)]$
- $\mathsf{Tree}(x(X).P) = \mathsf{db}(x)[\mathsf{Tree}(P)]$
- $\mathsf{Tree}(\overline{x}(Q)) = \mathsf{db}(x)^{O}[\mathsf{Tree}(Q)]$
- $\mathsf{Tree}(\Pi_{i=1}^{n} P_i) = \Pi_{i=1}^{n}[\mathsf{Tree}(P_1), \ldots, \mathsf{Tree}(P_n)]$
- $\mathsf{Tree}(\langle X\rangle P) = \mathsf{abs}[\mathsf{Tree}(P)]$
- $\mathsf{Tree}(\langle P\rangle Q) = \mathsf{app}[\mathsf{Tree}(P), \mathsf{Tree}(Q)]$
- $\mathsf{Tree}(\langle x\rangle P) = \mathsf{abs}[\mathsf{Tree}(P)]$
- $\mathsf{Tree}(\langle P\rangle n) = \mathsf{app}[\mathsf{Tree}(P), n[\,]]$

db assigns uniform indices to variables [De Bruijn, 1972; Lanese et al., 2011].

Tree normalization

Application: App(n_{raw}, ind, n_{eval})

Input: Tree nodes n_{raw}, n_{eval}, an integer ind.

Output: Tree nodes for which application is done.

if (n_{raw}.*type* == *'var'* **or** n_{raw}.*type* == *'inp'*) **and** n_{raw}.*label* == ind **then**
    $n_{raw} = n_{eval}$;
end if
if n_{raw}.*type* == *'out'* **and** n_{raw}.*label* == ind^O **then**
    $n_{raw} = n_{eval}$;
    n_{raw}.label = (n_{raw}.label)^O;
end if
if n_{raw}.*type* == *'inp'* **or** n_{raw}.*type* == *'abs'* **then**
    $ind = ind + 1$;
end if
for i = *1* **to** n.*numChildren* **do**
    App(n_{raw}.children[i], ind , n_{eval});
end for

Tree normalization

Normalization Step 1: NS1(n)

Input: A tree node n.

Output: Tree node after normalization step 1.

```
for i = 1 to n.numChildren do
    NS1(n.children[i]);
end for
if n.type == 'app' then
    if n.children[1].type == 'abs' then
        App(n.children[1].children[1], 1, n.children[2]);
    end if
end if
```

Tree normalization

Normalization Step 2: NS2(n)

Input: A tree node n.

Output: Tree node after normalization step 2.

```
for i = 1 to n.numChildren do  NS2(n.children[i]);
if n.type == 'par' then
    j = 1;
    for i = 1 to n.numChildren do
        if n.children[i].type ≠ 'zero' then
            n.children[j] = n.children[i];
            j = j + 1;
        end if
    end for
    n.numChildren = j - 1;
    if n.numChildren == 0 then
        n.type = 'zero';
    else if n.numChildren == 1 then
        n = n.children[1];
    end if
end if
sortChildren(n);
```

Tree normalization

Normalization Step 3: NS3(n)

Input: A tree node n.

Output: Tree node after normalization step 3.

```
for i = 1 to n.numChildren do  NS3(n.children[i]);
if n.type == 'inp' then
    p = n.children[1];
    if p.type == 'par' then
        smallIndex = -1;  small = null;  big = null;
        pc1 = p.children[1];  pc2 = p.children[p.numChildren];
        if pc1.type == 'inp' and pc1.label == n.label and pc1.children[1] == pc2 then
            small = pc2;  big = pc1;  smallIndex = p.numChildren;
        else if pc2.type=='inp' and pc2.label == n.label and pc2.children[1]==pc1 then
            small = pc1;  big = pc2;  smallIndex = 1;
        else  return;
        for i = 2 to n.numChildren-1 do
            if p.children[i] ≠ big then  return;
        end for
        p.children[smallIndex] = big;
        n = n.children[1];
    end if
end if
```

Complexity of the algorithm

Let n be the total number of nodes in the tree representations of the processes under consideration.

The bisimilarity checking algorithm has the following complexity.

Time $O(n \log(n))$.

Space $O(n)$.

4 Conclusion

1. In presence of parameterization, the strong bisimilarity is still decidable for Π^{mp}.
2. A complete axiom system for the strong bisimilarity is provided.
3. An algorithm for the bisimilarity checking is designed.

Future work

1. Expanding the model to allow more modelling capability, e.g., locations or probability, while maintaining the decidability result.
2. Considering the decidability of the weak bisimilarity.

Thank you.