Algebraic Process Calculi

Formalisms 18/19, v2

TU Berlin

Uwe Nestmann

August 19, 2019

Contents

1 Simulation and Bisimulation

1.1 Definition (Labeled Transition System / LTS)
Let $\mathcal{A}$ be an *alphabet of actions*.
An *LTS* over $\mathcal{A}$ is a pair $(\mathcal{Q}, \mathcal{T})$ with
- a set of *states* $\mathcal{Q} = \{ q_0, q_1, \dots \}$
- a ternary *transition relation* $\mathcal{T} \subseteq (\mathcal{Q} \times \mathcal{A} \times \mathcal{Q})$

A transition $(q, \mu, q') \in \mathcal{T}$ is also written $q \xrightarrow{\mu}_{\mathcal{T}} q'$.
If $q \xrightarrow{\mu_1}_{\mathcal{T}} q_1 \dots \xrightarrow{\mu_n}_{\mathcal{T}} q_n$ we call q_n a *derivative* of q.
Usually we omit the subscript $\mathcal{T}$ of arrows.

1.2 Definition ((Strong) Simulation)
Let $(\mathcal{Q}, \mathcal{T})$ be an LTS (over $\mathcal{A}$).
1. Let $S \subseteq \mathcal{Q} \times \mathcal{Q}$ be a binary relation.
   S is a *(strong) simulation* on $(\mathcal{Q}, \mathcal{T})$ if,
   whenever $p \; S \; q$,
   if $p \xrightarrow{\mu} p'$ (for some $p' \in \mathcal{Q}$) then
   there is $q' \in \mathcal{Q}$ such that $q \xrightarrow{\mu} q'$ and $p' \; S \; q'$.
   Formally this means:

$$\begin{aligned} &\forall p, q \in \mathcal{Q} \,.\, p \; S \; q \\ &\quad \to \forall p' \in \mathcal{Q} \,.\, \forall \mu \in \mathcal{A} \,.\, p \xrightarrow{\mu} p' \\ &\qquad\qquad \to \exists q' \in \mathcal{Q} \,.\, q \xrightarrow{\mu} q' \wedge p' \; S \; q' \end{aligned}$$

2. *q (strongly) simulates p*, written $p \preceq q$,
   if there is a (strong) simulation S (on $(\mathcal{Q}, \mathcal{T})$)
   such that $p \; S \; q$.

The relation $\preceq \subseteq \mathcal{Q} \times \mathcal{Q}$ is also called *similarity*.

1.3 Notation
In the following, if the used transition system is clear in the respective reasoning context, then we usually omit to specify "on $\mathcal{Q}$" (for binary relations) or "on $(\mathcal{Q}, \mathcal{T})$" (for simulations) and even the alphabet of actions $\mathcal{A}$.

1.4 Lemma
Let $(\mathcal{Q}, \mathcal{T})$ be an LTS.
If S_1 and S_2 are simulations, then
1. $S_1 \cup S_2$ is also a simulation.
2. $S_1 S_2$ is also a simulation.
3. $S_1 \cap S_2$ is not necessarily a simulation.

1.5 Proposition
Let $(\mathcal{Q}, \mathcal{T})$ be an LTS.
1. $\preceq = \bigcup\{ \ S \mid S \text{ is a simulation on } (\mathcal{Q}, \mathcal{T}) \ \}$
2. $\preceq \subseteq \mathcal{Q} \times \mathcal{Q}$ is the largest simulation on $(\mathcal{Q}, \mathcal{T})$.
3. $\preceq \subseteq \mathcal{Q} \times \mathcal{Q}$ is a preorder.

1.6 Definition (Mutual Simulation)
Let $(\mathcal{Q}, \mathcal{T})$ be an LTS. Let $p, q \in \mathcal{Q}$.
p and q are *mutually similar*, written $p \eqslantgtr q$,
if there is a pair (S_1, S_2) of simulations S_1 and S_2
with $p \ S_1 \ q \ S_2 \ p$ (i.e., with $p \ S_1 \ q$ and $q \ S_2 \ p$).

1.7 Proposition
$\eqslantgtr$ is an equivalence relation.

1.8 Definition ((Strong) Bisimulation)
Let $(\mathcal{Q}, \mathcal{T})$ be an LTS.
A binary relation B on $\mathcal{Q}$ is
a (strong) bisimulation on $(\mathcal{Q}, \mathcal{T})$
if both B and B^{-1} are (strong) simulations.
p and q are *(strongly) bisimilar*, written $p \sim q$,
if there is a (strong) bisimulation B such that $p \ B \ q$.

1.9 Proposition
1. $\sim = \bigcup\{ \ S | S \text{ is a (strong) bisimulation on } (\mathcal{Q}, \mathcal{T}) \ \}$
2. $\sim \subseteq \mathcal{Q} \times \mathcal{Q}$ is the largest bisimulation on $(\mathcal{Q}, \mathcal{T})$.
3. $\sim \subseteq \mathcal{Q} \times \mathcal{Q}$ is an equivalence relation.

2 Sequential Processes

2.1 Notation

We use the following sets of entities with the corresponding meta-variables:

$\mathcal{I}$	*process identifiers*	$\mathsf{A}, \mathsf{B}, \ldots$
$\mathcal{N}$	*names*	$a, b, c, \ldots$
$\overline{\mathcal{N}}$	*co-names*	$\overline{a}, \overline{b}, \overline{c}, \ldots$
$\mathcal{L}$	*labels*	$\lambda, \ldots \in \mathcal{L} \stackrel{\text{def}}{=} \mathcal{N} \cup \overline{\mathcal{N}}$
$\mathcal{A}$	*actions*	$\mu, \beta, \ldots \in \mathcal{L} \cup \{ \tau \}$

Labels are often also called *visible/external* actions.
In contrast, τ is called *invisible/internal* action.

We use $\vec{a}$ to denote *finite sequences* $(a_1, \ldots, a_n)$ of *names*;
we sometimes use vectors without enclosing parentheses;
we use $\{ \vec{a} \}$ to denote the corresponding set $\{a_1, \ldots, a_n\}$.
A vector $\vec{a}$ is called *duplicate-free* if $|\vec{a}| = |\{\vec{a}\}|$.

We will use *parameterized processes* $\mathsf{A}\langle \vec{a} \rangle$ with *name* parameters (not, e.g., *co*-names or τ).

2.2 Definition (Sequential Process Expressions)

The sets $\mathcal{P}^{\text{seq}}$ and $\mathcal{M}^{\text{seq}}$ of sequential process expressions is defined by the following BNF-syntax (as generated by the non-terminals P and M, respectively):

$$\begin{aligned} P &::= \mathsf{A}\langle a_1, \ldots, a_n \rangle \mid M \\ M &::= \mathbf{0} \mid \mu.P \mid M + M \end{aligned}$$

We use $P, P_i, \ldots$ to denote *process expressions*,
while any $M, M_i, \ldots$ always denotes a *choice* or *sum[mation]*.
Any summation not containing prefixes $\mu.P$ (i.e., only $\mathbf{0}$) is called *empty summation*.

Each process identifier A that is used in a process expression is assumed to have a type-conforming *defining equation*

$$\mathsf{A}(\, a_1, \ldots, a_n \,) := M$$

(note the use of parentheses in this case), where 1) M is a summation, 2) $(a_1, \ldots, a_n)$ is duplicate-free, 3) $\mathrm{fn}(M) \subseteq \{a_1, \ldots, a_n\}$, i.e., the *(free) names* (see Definition 2.3) of M appear in the list of parameters, and 4) $\{a_1, \ldots, a_n\} \subseteq \mathcal{N}$, i.e., only names are allowed as parameters, not co-names, nor τ.

For convenience, with $\vec{a} = (a_1, \ldots, a_n)$ and $\vec{b} = (b_1, \ldots, b_m)$, we write $\mathsf{A}(\, \vec{a} \,)$ and $\mathsf{A}\langle\, \vec{a} \,\rangle$ to denote $\mathsf{A}(\, a_1, \ldots, a_n \,)$ and $\mathsf{A}\langle\, a_1, \ldots, a_n \,\rangle$. Moreover, we write $\mathsf{A}(\, \vec{a}, \vec{b} \,)$ to denote $\mathsf{A}(\, a_1, \ldots, a_n, b_1, \ldots, b_m \,)$. Analogously, we use $\mathsf{A}\langle\, \vec{a}, \vec{b} \,\rangle$ to denote $\mathsf{A}\langle\, a_1, \ldots, a_n, b_1, \ldots, b_m \,\rangle$.

If $\mathsf{A}(\, \vec{a} \,) := M$ is given, then $\mathsf{A}\langle\, \vec{b} \,\rangle$ "behaves like" $\{{}^{\vec{b}}/_{\vec{a}}\}M$, which is defined as the simultaneous substitution of all occurrences of names in $\vec{a}$ by names in $\vec{b}$ (see Definition 2.4 and Notation 2.5).

2.3 Definition ((Free) Names)
$\mathrm{fn} : \mathcal{P}^{\mathrm{seq}} \to 2^{\mathcal{N}}$
The set $\mathrm{fn}(P)$ is defined inductively by:

$$\mathrm{fn}(\mu) \stackrel{\text{def}}{=} \begin{cases} \{\, b \,\} & \text{if } \mu = b \\ \{\, b \,\} & \text{if } \mu = \bar{b} \\ \emptyset & \text{if } \mu = \tau \end{cases}$$

$$\mathrm{fn}(\mathbf{0}) \stackrel{\text{def}}{=} \emptyset$$

$$\mathrm{fn}(\mu.P) \stackrel{\text{def}}{=} \mathrm{fn}(\mu) \cup \mathrm{fn}(P)$$

$$\mathrm{fn}(M_1 + M_2) \stackrel{\text{def}}{=} \mathrm{fn}(M_1) \cup \mathrm{fn}(M_2)$$

$$\mathrm{fn}(\mathsf{A}\langle\, \vec{a} \,\rangle) \stackrel{\text{def}}{=} \{\, \vec{a} \,\}$$

2.4 Definition (Simultaneous Substitution)
A substitution σ is a total function $\sigma : \mathcal{N} \to \mathcal{N}$.
The set $\sup(\sigma) \stackrel{\text{def}}{=} \{\, n \in \mathcal{N} \mid \sigma(n) \neq n \,\}$
denotes the *support* of σ.

1. For $k \in \mathbb{N}$, we lift σ to $\mathcal{N}^k \to \mathcal{N}^k$
   to act on vectors of names by

$$\sigma(\,(n_1, \ldots, n_k)\,) \stackrel{\text{def}}{=} (\,\sigma(n_1), \ldots, \sigma(n_k)\,)$$

2. We lift σ to actions $\mathcal{A} \to \mathcal{A}$, as defined by:

$$\sigma(\mu) \stackrel{\text{def}}{=} \begin{cases} a' & \text{if } \mu = a \in \mathcal{N} \text{ and } \sigma(a) = a' \\ \overline{\sigma(a)} & \text{if } \mu = \overline{a} \\ \tau & \text{if } \mu = \tau \end{cases}$$

3. We lift σ to processes $\mathcal{P}^{\text{seq}} \to \mathcal{P}^{\text{seq}}$,
   as inductively defined by:

$$\begin{aligned} \sigma(\mathbf{0}) &\stackrel{\text{def}}{=} \mathbf{0} \\ \sigma(\mu.P) &\stackrel{\text{def}}{=} \sigma(\mu).\sigma(P) \\ \sigma(M_1 + M_2) &\stackrel{\text{def}}{=} \sigma(M_1) + \sigma(M_2) \\ \sigma(\mathsf{A}\langle\, \vec{a} \,\rangle) &\stackrel{\text{def}}{=} \mathsf{A}\langle\, \sigma(\vec{a}) \,\rangle \end{aligned}$$

2.5 Notation
Let $\sigma : \mathcal{N} \to \mathcal{N}$ be a substitution with finite support. We often represent σ as $\{{}^{\sigma(\vec{n})}/_{\vec{n}}\}$, where $\vec{n}$ is an arbitrary vector enumerating the elements of $\sup(\sigma)$.
Dually, given any vectors $\vec{a}$ and $\vec{b}$ of equal length, then $\{{}^{\vec{b}}/_{\vec{a}}\}$ uniquely defines a substitution, if $\vec{a}$ is duplicate-free.

2.6 Definition (Operational Semantics)
The LTS $(\mathcal{P}^{\mathrm{seq}}, \mathcal{T})$ of sequential process expressions over $\mathcal{A}$ has $\mathcal{P}^{\mathrm{seq}}$ as states, and its transitions $\mathcal{T}$ are generated by the following rules:

$$\textsc{pre:}\ \frac{}{\mu.P \xrightarrow{\mu} P} \qquad \textsc{def:}\ \frac{\{\vec{c}/\vec{a}\}M \xrightarrow{\mu} P'}{\mathsf{A}\langle \vec{c} \rangle \xrightarrow{\mu} P'}\ \mathsf{A}(\vec{a}) := M$$

$$\textsc{sum}_1\textsc{:}\ \frac{M_1 \xrightarrow{\mu} M_1'}{M_1 + M_2 \xrightarrow{\mu} M_1'} \qquad \textsc{sum}_2\textsc{:}\ \frac{M_2 \xrightarrow{\mu} M_2'}{M_1 + M_2 \xrightarrow{\mu} M_2'}$$

Note that "transition under prefix" is not allowed.

2.7 Notation
We also use the abbreviation

$$\sum_{i \in I} \mu_i.P_i \stackrel{\mathrm{def}}{=} \begin{cases} \mathbf{0} & \text{if } I = \emptyset \\ \mu_k.P_k + \sum_{i \in (I \setminus \{k\})} \mu_i.P_i & \text{otherwise} \end{cases}$$

where

- I is a (usually finite) indexing set, and
- $k \in I$ is selected according to some implicit ordering or, if none exists, by the axiom of choice.

The order of components is not semantically relevant.

We also use the notation

$$\sum \mathbf{M} \stackrel{\mathrm{def}}{=} \sum_{M \in \mathbf{M}} M.$$

where $\mathbf{M}$ is a finite subset of $\mathcal{M}^{\mathrm{seq}}$.

3 Concurrent Processes

3.1 Definition (Concurrent Process Expressions)
The sets $\mathcal{P}$ and $\mathcal{M}$ of concurrent process expressions is defined by the following BNF-syntax:

$$\begin{array}{rcl} P & ::= & \mathsf{A}\langle \vec{a} \rangle \mid M \mid P|P \mid (\nu a)\, P \\ M & ::= & \mathbf{0} \mid \mu.P \mid M + M \end{array}$$

under the same assumptions on process identifiers as in the sequential case. The expression $P|P$ denotes *parallel composition*, while $(\nu a)\, P$ denotes *name generation*, also called *restriction*.

3.2 Notation If necessary, we use parentheses to clarify the scope of the various process operators in process expressions. Moreover, we impose that unary operators have precedence over (i.e., bind tighter than) binary operators. From the definition of the syntax we have that $+$ binds stronger than $|$. E.g., $\mathbf{0} + (\mathbf{0}|\mathbf{0})$ is not an element of $\mathcal{P}$.

$$\begin{array}{rcl} (\nu a)\, P \mid Q & = & ((\nu a)\, P) \mid Q \\ a.P + M & = & (a.P) + M \\ \sigma M_1 + M_2 & = & \sigma(M_1) + M_2 \end{array}$$

3.3 Notation
We allow the shorthands of Notation 2.7 also for $\mathcal{P}$ and $\mathcal{M}$. Similarly, we use the abbreviation

$$\prod_{i \in I} P_i \stackrel{\text{def}}{=} \begin{cases} \mathbf{0} & \text{if } I = \emptyset \\ P_k \mid \prod_{i \in (I \setminus \{k\})} P_i & \text{otherwise} \end{cases}$$

where

- I is a (usually finite) indexing set, and
- $k \in I$ is selected according to some implicit ordering or, if none exists, by the axiom of choice.

The order of components is not semantically relevant.

3.4 Definition (Free Names)
$\mathrm{fn} : \mathcal{P} \to 2^{\mathcal{N}}$.
The set $\mathrm{fn}(P)$ is defined inductively by:

$$\mathrm{fn}(\mu) \stackrel{\text{def}}{=} \begin{cases} \{ b \} & \text{if } \mu = b \\ \{ b \} & \text{if } \mu = \bar{b} \\ \emptyset & \text{if } \mu = \tau \end{cases}$$

$$\begin{aligned} \mathrm{fn}(\mathbf{0}) &\stackrel{\text{def}}{=} \emptyset \\ \mathrm{fn}(\mu.P) &\stackrel{\text{def}}{=} \mathrm{fn}(\mu) \cup \mathrm{fn}(P) \\ \mathrm{fn}(M_1 + M_2) &\stackrel{\text{def}}{=} \mathrm{fn}(M_1) \cup \mathrm{fn}(M_2) \\ \mathrm{fn}(\mathsf{A}\langle \vec{a} \rangle) &\stackrel{\text{def}}{=} \{ \vec{a} \} \\ \mathrm{fn}(P_1 | P_2) &\stackrel{\text{def}}{=} \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2) \\ \mathrm{fn}((\boldsymbol{\nu} a)\, P) &\stackrel{\text{def}}{=} \mathrm{fn}(P) \setminus \{ a \} \end{aligned}$$

We say that *a occurs free* in P, if $a \in \mathrm{fn}(P)$.

3.5 Definition (Bound Names)
$\mathrm{bn} : \mathcal{P} \to 2^{\mathcal{N}}$
The set $\mathrm{bn}(P)$ is defined inductively by:

$$\mathrm{bn}(\mu) \stackrel{\text{def}}{=} \begin{cases} \emptyset & \text{if } \mu = b \\ \emptyset & \text{if } \mu = \bar{b} \\ \emptyset & \text{if } \mu = \tau \end{cases}$$

$$\mathrm{bn}(\mathbf{0}) \stackrel{\text{def}}{=} \emptyset$$

$$\mathrm{bn}(\mu.P) \stackrel{\text{def}}{=} \mathrm{bn}(\mu) \cup \mathrm{bn}(P)$$

$$\mathrm{bn}(M_1 + M_2) \stackrel{\text{def}}{=} \mathrm{bn}(M_1) \cup \mathrm{bn}(M_2)$$

$$\mathrm{bn}(\mathsf{A}\langle \vec{a} \rangle) \stackrel{\text{def}}{=} \emptyset$$

$$\mathrm{bn}(P_1|P_2) \stackrel{\text{def}}{=} \mathrm{bn}(P_1) \cup \mathrm{bn}(P_2)$$

$$\mathrm{bn}((\boldsymbol{\nu} a)\, P) \stackrel{\text{def}}{=} \mathrm{bn}(P) \cup \{\, a \,\}$$

We say that *a occurs bound* in P,
if P has a subterm $(\boldsymbol{\nu} a)\, Q$, where a occurs free in Q.
We say $(\boldsymbol{\nu} a)\, P$ *binds* (any free occurrence of) a in P.

For example, a does *not occur* bound in $(\boldsymbol{\nu} a)\, \mathbf{0}$,
although $a \in \mathrm{bn}((\boldsymbol{\nu} a)\, \mathbf{0})$.

3.6 Definition
A name a is called *fresh* with respect to an expression P, if $a \notin \mathrm{fn}(P) \cup \mathrm{bn}(P)$.

3.7 Definition (α-conversion)
The process P' is a *simple α-conversion* of P if it can be obtained by replacing an instance of a subterm $(\nu a)\, Q$ of P with $(\nu b)\, Q'$, where b is fresh with respect to Q, and where Q' is obtained by replacing all free occurrences of a in Q with b.

The relation $=_\alpha$ (of type $\mathcal{P} \times \mathcal{P}$), called *$\alpha$-congruence*, is the smallest equivalence relation containing simple α-conversion.

3.8 Definition ((name-)clash-free)
A process $P \in \mathcal{P}$ is called *[name-]clash-free* (or *α-free*) if $\mathrm{fn}(P) \cap \mathrm{bn}(P) = \emptyset$ and all set unions in the definition of $\mathrm{bn}(P)$ are disjoint unions (i.e., the same name is not bound twice in P).

3.9 Lemma
For every expression $P \in \mathcal{P}$, there exists a clash-free expression $\widehat{P} \in \mathcal{P}$ such that $P =_\alpha \widehat{P}$.
In this case, we call $\widehat{P}$ a *clash-free version* of P.

3.10 Definition (Simultaneous Substitution)
Any substitution $\sigma : \mathcal{N} \to \mathcal{N}$ is lifted to concurrent process expressions $\mathcal{P} \to \mathcal{P}$, inductively defined by:

$$\begin{aligned}
\sigma(\mathbf{0}) &\stackrel{\text{def}}{=} \mathbf{0} \\
\sigma(\mu.P) &\stackrel{\text{def}}{=} \sigma(\mu).\sigma(P) \\
\sigma(M_1 + M_2) &\stackrel{\text{def}}{=} \sigma(M_1) + \sigma(M_2) \\
\sigma(\mathsf{A}\langle\, \vec{a}\,\rangle) &\stackrel{\text{def}}{=} \mathsf{A}\langle\, \sigma(\vec{a})\,\rangle \\
\sigma(P_1|P_2) &\stackrel{\text{def}}{=} \sigma(P_1) \mid \sigma(P_2) \\
\sigma((\boldsymbol{\nu} a)\, P) &\stackrel{\text{def}}{=} (\boldsymbol{\nu} a)\, (\sigma[a \mapsto a])(P)
\end{aligned}$$

where

$$\sigma[a \mapsto a](x) \stackrel{\text{def}}{=} \begin{cases} a & \text{if } x = a \\ \sigma(x) & \text{if } x \neq a \end{cases}$$

We say that σ *avoids name-clashes* on P (written $\mathrm{cf}(\sigma, P)$), if $\sigma(\mathrm{sup}(\sigma)) \cap \mathrm{bn}(P) = \emptyset$.

The use of substitutions that avoid name-clashes guarantees that any occurrence of a free name is not in danger to become bound after having been replaced (e.g., $\{^{a}/_{b}\}(\boldsymbol{\nu} a)\, b.\mathbf{0} = (\boldsymbol{\nu} a)\, a.\mathbf{0}$).

To ensure that we always avoid name-clashes when applying substitutions, we "silently assume" that an appropriate α-conversion is implicitly applied to expressions whenever necessary.

3.11 Definition (Operational Semantics)
The LTS $(\mathcal{P}, \mathcal{T})$ of concurrent process expressions $\mathcal{P}$ over actions $\mathcal{A}$ has transitions $\mathcal{T}$ generated by the following rules:

$$\textsc{pre}: \frac{}{\mu.P \xrightarrow{\mu} P} \qquad \textsc{def}: \frac{\{\vec{c}/\vec{a}\}M' \xrightarrow{\mu} P'}{\mathsf{A}\langle \vec{c} \rangle \xrightarrow{\mu} P'} \Phi$$

$$\Phi \stackrel{\text{def}}{=} \mathsf{A}(\vec{a}) := M \text{ AND } M =_\alpha M' \text{ AND } \mathrm{cf}(\{\vec{c}/\vec{a}\}, M')$$

$$\textsc{sum}_1: \frac{M_1 \xrightarrow{\mu} M_1'}{M_1 + M_2 \xrightarrow{\mu} M_1'} \qquad \textsc{sum}_2: \frac{M_2 \xrightarrow{\mu} M_2'}{M_1 + M_2 \xrightarrow{\mu} M_2'}$$

$$\textsc{par}_1: \frac{P_1 \xrightarrow{\mu} P_1'}{P_1 | P_2 \xrightarrow{\mu} P_1' | P_2} \qquad \textsc{par}_2: \frac{P_2 \xrightarrow{\mu} P_2'}{P_1 | P_2 \xrightarrow{\mu} P_1 | P_2'}$$

$$\textsc{com}: \frac{P \xrightarrow{\lambda} P' \qquad Q \xrightarrow{\overline{\lambda}} Q'}{P | Q \xrightarrow{\tau} P' | Q'}$$

$$\textsc{res}: \frac{P \xrightarrow{\mu} P'}{(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'} \; \mu \notin \{a, \overline{a}\}$$

where $\overline{\overline{\lambda}} \stackrel{\text{def}}{=} \lambda$.

NB: The side condition Φ of rule DEF induces, in principle, an infinite number of transitions, which only differ in the choice of bound names, as introduced by $M =_\alpha M'$. For simplicity, we only count these as a single transition; in other words, we then work with an LTS that considers representatives of α-congruence classes of $\mathcal{P}$ instead of just $\mathcal{P}$.

3.12 Proposition
For each $P \in \mathcal{P}$, there is a finite index set I, and for all $i \in I$ there are actions μ_i and processes Q_i such that

$$P \sim \sum_{i \in I} \{\ \mu_i.Q_i \mid P \xrightarrow{\mu_i} Q_i\ \}.$$

3.13 Proposition
For all $n \geq 0$ and $P_1, \ldots, P_n \in \mathcal{P}$:

$$P_1|\ldots|P_n \sim \left\{ \begin{array}{l} \sum \{\, \beta.(\ P_1|\ldots|P_i'|\ldots|P_n\ ) \\ \qquad \mid \exists\, 1 \leq i \leq n. P_i \xrightarrow{\beta} P_i'\ \} \\ + \\ \sum \{\, \tau.(\ P_1|\ldots|P_i'|\ldots|P_j'|\ldots|P_n\ ) \\ \qquad \mid \exists\, 1 \leq i < j \leq n. P_i \xrightarrow{\lambda} P_i' \wedge P_j \xrightarrow{\overline{\lambda}} P_j'\ \} \end{array} \right.$$

3.14 Proposition
For all $n \geq 0$, $P_1, \ldots, P_n \in \mathcal{P}$, and $\vec{a}$:

$$(\nu\vec{a})\,(\ P_1|\ldots|P_n\ ) \sim \left\{ \begin{array}{l} \sum \{\, \beta.(\nu\vec{a})\,(\ P_1|\ldots|P_i'|\ldots|P_n\ ) \\ \qquad \mid \exists\, 1 \leq i \leq n. P_i \xrightarrow{\beta} P_i' \wedge \beta, \overline{\beta} \notin \vec{a}\ \} \\ + \\ \sum \{\, \tau.(\nu\vec{a})\,(\ P_1|\ldots|P_i'|\ldots|P_j'|\ldots|P_n\ ) \\ \qquad \mid \exists\, 1 \leq i < j \leq n. P_i \xrightarrow{\lambda} P_i' \wedge P_j \xrightarrow{\overline{\lambda}} P_j'\ \} \end{array} \right.$$

4 Congruence and Reaction

4.1 Notation
We often write $(\nu ab)\, P$ instead of $(\nu a)\,(\nu b)\, P$.
We often omit trailing $.\mathbf{0}$ and abbreviate $a.\mathbf{0}$ by a.

4.2 Definition (Process Contexts)
A *process context* $C[\cdot]$ is
defined by the following syntax:

$$\begin{aligned} C[\cdot] \;::=\; & [\cdot] \mid \mu.C[\cdot] + M \mid M + \mu.C[\cdot] \\ & \mid C[\cdot] \end{aligned}$$

The *elementary process contexts* are

$$(\nu a)\,[\cdot] \qquad \mu.[\cdot] + M \qquad M + \mu.[\cdot]$$
$$[\cdot] \mid P \qquad P \mid [\cdot]$$

The expression $C[Q]$ denotes the result of
filling the hole $[\cdot]$ of $C[\cdot]$ with process Q.

4.3 Definition (Process Congruence)
Let $\cong$ be an equivalence relation over $\mathcal{P}$.
Then, $\cong$ is said to be a process congruence,
if, for *all* process contexts $C[\cdot]$, $P \cong Q$ implies $C[P] \cong C[Q]$.

4.4 Proposition
Let $\cong$ be an equivalence relation over $\mathcal{P}$.
Then, $\cong$ is a process congruence,
if and only if it is preserved by *all elementary* contexts,
i.e., if $P \cong Q$ implies all of the following:

$$\mu.P + M \cong \mu.Q + M \qquad P|R \cong Q|R$$
$$M + \mu.P \cong M + \mu.Q \qquad R|P \cong R|Q$$
$$(\nu a)\, P \cong (\nu a)\, Q$$

for all possible $\mu \in \mathcal{A}$, $a \in \mathcal{N}$, $R \in \mathcal{P}$ and $M \in \mathcal{M}$.

4.5 Theorem
Bisimilarity $\sim \subseteq \mathcal{P} \times \mathcal{P}$ is a process congruence.

4.6 Definition (Structural Congruence)
Structural Congruence, written $\equiv$, is the smallest process congruence over $\mathcal{P}$ containing the following equations.

1. $=_\alpha$
2. commutative monoid laws for $(\mathcal{M}, +, \mathbf{0})$
3. commutative monoid laws for $(\mathcal{P}, \mid, \mathbf{0})$
4. rules for restriction:

$$(\boldsymbol{\nu} a)\,(P \mid Q) \equiv P \mid (\boldsymbol{\nu} a)\,Q, \quad \text{if } a \notin \mathrm{fn}(P)$$
$$(\boldsymbol{\nu} ab)\,P \equiv (\boldsymbol{\nu} ba)\,P$$
$$(\boldsymbol{\nu} a)\,\mathbf{0} \equiv \mathbf{0}$$

5. $\mathsf{A}\langle \vec{b} \rangle \equiv \{^{\vec{b}}/_{\vec{a}}\} M'_{\mathsf{A}}$,
   if $\mathsf{A}(\vec{a}) := M_{\mathsf{A}}$ and $M_{\mathsf{A}} =_\alpha M'_{\mathsf{A}}$ and $\mathrm{cf}(\{^{\vec{b}}/_{\vec{a}}\}, M'_{\mathsf{A}})$.

Note that n-ary summation/product is already implicitly defined modulo the commutative monoid laws.

There are at least four usages for structural congruence.

The first usage of structural congruence is that process descriptions can be transformed into a shape that allows to immediately assess all top-level parallel components.

4.7 Definition (Standard Form)
A process expression $(\boldsymbol{\nu}\vec{a})\,(\prod_{i \in I} M_i)$,
where each M_i is a non-empty sum,
is said to be in *standard form*.
If $\{\vec{a}\}$ is empty then there is no restriction.

4.8 Theorem
Every process is structurally congruent to some standard form.

The second usage of structural congruence is a much simpler presentation of operational semantics that focuses on handshakes.

4.9 Definition (Reaction Semantics)
The relation $\mapsto$ over $\mathcal{P}$ is generated by the following rules:

$$\textsc{tau:}\ \frac{}{\tau.P+M \mapsto P} \qquad \textsc{react:}\ \frac{}{a.P+M \mid \overline{a}.Q+N \mapsto P \mid Q}$$

$$\textsc{par:}\ \frac{P \mapsto P'}{P \mid Q \mapsto P' \mid Q} \qquad \textsc{res:}\ \frac{P \mapsto P'}{(\nu a)\, P \mapsto (\nu a)\, P'}$$

$$\textsc{struct:}\ \frac{P \mapsto P'}{Q \mapsto Q'}\ P \equiv Q \text{ and } P' \equiv Q'$$

4.10 Proposition
If $P \xrightarrow{\mu} P'$ and $P \equiv Q$,
then there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \equiv Q'$.

4.11 Corollary
$\equiv$ is a strong bisimulation.

4.12 Theorem
$P \mapsto P'$ iff $P \xrightarrow{\tau} \equiv P'$.

The third usage of structural congruence is a proof technique for bisimilarity.

4.13 Definition (Strong Simulation up to $\equiv$)
A binary relation S on $\mathcal{P}$ is
a *(strong) simulation up to* $\equiv$
if, whenever $P\ S\ Q$, then if $P \xrightarrow{\mu} P'$ then
there is $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\mu} Q'$ and $P' \equiv S \equiv Q'$.
S is a strong bisimulation up to $\equiv$
if its converse also has this property.

4.14 Proposition
If S is a (strong) bisimulation up to $\equiv$ and $P \; S \; Q$,
then $P \sim Q$.

The fourth usage of structural congruence is to statically check for currently observable actions without referring to a labeled transition semantics.

4.15 Definition (Barbs)
Given the LTS $(\mathcal{P}, \mapsto)$.
Let $P, Q \in \mathcal{P}$ and $\lambda \in \mathcal{L}$.
Let $\Mapsto \stackrel{\text{def}}{=} \mapsto^*$ (reflexive-transitive closure of $\mapsto$).
Then:

1. Q has a *strong barb* for λ, written $Q \downarrow_\lambda$, if

$$Q \equiv (\boldsymbol{\nu} \vec{a}) \, (\; Q_1 \mid \lambda.Q_2 + M \;)$$

   for some $Q_1, Q_2 \in \mathcal{P}$, $M \in \mathcal{M}$
   and $\{\vec{a}\} \subseteq \mathcal{N}$ with $\{\lambda, \overline{\lambda}\} \cap \{\vec{a}\} = \emptyset$.

2. P has a *weak barb* for λ, written $P \Downarrow_\lambda$,
   if $P \Mapsto Q$ for some $Q \in \mathcal{P}$ with $Q \downarrow_\lambda$.

4.16 Proposition
Let $P \in \mathcal{P}$ and $\lambda \in \mathcal{L}$.
Then $P \downarrow_\lambda$ iff $P \xrightarrow{\lambda} P'$ for some $P' \in \mathcal{P}$.

4.17 Definition (Barbed congruence)
Barbed bisimilarity on $\mathcal{P}$ is the largest symmetric relation $\approx_{\mathrm{b}} \subseteq \mathcal{P} \times \mathcal{P}$ such that, whenever $P \approx_{\mathrm{b}} Q$, then

1. $P \downarrow_\lambda$ implies $Q \Downarrow_\lambda$, and
2. $P \mapsto P'$ implies $Q \Mapsto Q'$ with $P' \approx_{\mathrm{b}} Q'$.

Processes P and Q are *barbed congruent*, written $P \cong^{\mathrm{c}} Q$, if $C[P] \approx_{\mathrm{b}} C[Q]$ for every context $C[\cdot]$.

5 Observation Equivalence

5.1 Definition (Weak Transitions)
Given the LTS $(\mathcal{P}, \rightarrow)$ over $\mathcal{A}$.
Then, *weak* transitions are defined by:

1. $\Rightarrow \stackrel{\text{def}}{=} \stackrel{\tau}{\longrightarrow}^*$
2. $\stackrel{\mu}{\Longrightarrow} \stackrel{\text{def}}{=} \Rightarrow \stackrel{\mu}{\longrightarrow} \Rightarrow$

5.2 Definition (Weak Simulation)
Let S be a binary relation over $\mathcal{Q}$.
Then S is said to be a *weak simulation*
if, whenever $P\ S\ Q$,

- if $P \stackrel{\tau}{\longrightarrow} P'$ then there is $Q' \in \mathcal{Q}$ such that $Q \Rightarrow Q'$ and $P'\ S\ Q'$.
- if $P \stackrel{\lambda}{\longrightarrow} P'$ then there is $Q' \in \mathcal{Q}$ such that $Q \stackrel{\lambda}{\Longrightarrow} Q'$ and $P'\ S\ Q'$.

Q *weakly simulates* P,
if there is a weak simulation S such that $P\ S\ Q$.

5.3 Lemma
Every strong simulation is also a weak simulation.

5.4 Definition (Weak Bisimulation)
A binary relation B is *a weak bisimulation*
if both B and its converse B^{-1} are weak simulations.
P and Q are *weakly bisimilar*, *weakly equivalent*,
or *observation equivalent*, written $P \approx Q$,
if there exists a weak bisimulation B with $P\ B\ Q$.

5.5 Proposition
$P \sim Q$ implies $P \approx Q$.

5.6 Proposition (Weak Equivalence)

1. $\approx = \bigcup\{\ B \mid B \text{ is a weak bisimulation } \}$
2. $\approx$ is itself a weak bisimulation.
3. $\approx$ is an equivalence relation.

5.7 Theorem

Weak equivalence $\approx \subseteq \mathcal{P} \times \mathcal{P}$ is a process congruence.

5.8 Proposition

Let $P, Q \in \mathcal{P}$. Then $P \approx Q$ iff $P \cong^{c} Q$.

5.9 Proposition

For concurrent processes we have:

$$\Delta \subset =_{\alpha} \subset \equiv \subset \sim \subset \approx = \cong^{c} \subset \approx_{b} \subset \nabla$$

5.10 Definition (Weak simulation up to $\sim$)
S is a *weak simulation up to* $\sim$
if, whenever $P \; S \; Q$,

- if $P \xrightarrow{\tau} P'$ then there is $Q' \in \mathcal{Q}$ such that $Q \Rightarrow Q'$ and $P' \sim S \sim Q'$.
- if $P \xrightarrow{\lambda} P'$ then there is $Q' \in \mathcal{Q}$ such that $Q \stackrel{\lambda}{\Longrightarrow} Q'$ and $P' \sim S \sim Q'$.

S is a *weak bisimulation up to* $\sim$
if its converse also has this property.

5.11 Proposition
If B is a (weak) bisimulation up to $\sim$ and $P \; B \; Q$,
then $P \approx Q$.

5.12 Theorem (Unique Solution of Equations)
Assume a possibly infinite list of equations:

$$X_1 \approx \lambda_{11}.Y_{11} + \ldots + \lambda_{1n_1}.Y_{1n_1}$$
$$X_2 \approx \lambda_{21}.Y_{21} + \ldots + \lambda_{2n_2}.Y_{2n_2}$$
$$\ldots \approx \ldots$$

The $X_i, Y_{i,j}$ are process variables (i.e., not necessarily process constants) and every Y_{ij} is equal to some X_k.
If $\lambda_{ij} \neq \tau$ for all i, j, then there is a (up to $\approx$) unique sequence $P_1, P_2, \ldots$ of processes that satisfies the equations (i.e., $X_1 = P_1$, $X_2 = P_2, \ldots$ and all the equations hold).

6 Value-Passing

6.1 Notation

We use the following sets of entities with corresponding meta-variables:

$\mathcal{I}$	*process identifiers*	$\mathsf{A}, \mathsf{B}, \ldots$
$\mathcal{N}$	*(channel) names*	$a, b, c, \ldots$
$\mathcal{V}$	*values*	v, w
$\mathcal{B}$	*boolean expressions*	$e ::= \rho{=}\rho \mid \rho{<}\rho \mid e \wedge e \mid \neg e$
		$\rho ::= v \mid x$
$\mathcal{X}$	*variables*	x, y, z
$\mathcal{A}$	*actions*	$\mu ::= \overline{a}\langle x \rangle \mid \overline{a}\langle v \rangle \mid a(x) \mid \tau$
$\mathcal{L}$	*labels*	$\delta ::= \overline{a}\langle v \rangle \mid av \mid \tau$

"negative" actions $\overline{a}\langle v \rangle$: *send value* v *over channel* a.

"positive" actions $a(x)$: *receive any value, say* v, *over channel* a *and "bind" it to variable* x, *resulting in a* substitution $\{^{v}/_{x}\}$ *of the formal parameter* x *by the actual parameter* v.

Substitution $\{^{v}/_{x}\}$ is of values for variables, as opposed to names for names as in Definitions 2.4 and 3.10. We omit the straightforward definition. *Evaluation* requires a function $\mathcal{E} : \mathcal{B} \to \{ t, f \}$ that provides the usual interpretation of closed (i.e., variable-free) boolean expressions; for non-closed (so-called open) expressions ρ, $\mathcal{E}(e)$ returns f such that a term depending on such a ρ will not have any transition.

We could have chosen to allow for non-value expressions in output prefixes, e.g., $\overline{a}\langle e \rangle$ with e being an expression in an arbitrary programming language that can be evaluated to a proper value in finite time. One would then need an *evaluation strategy* telling that expressions-to-be-sent must, for example, be evaluated before being sent or be sent unevaluated. However, we keep the language as simple as possible in order to put the focus on the effect of bisimulation (see Definitions 6.7 and 6.8, which is already visible when communicating just values.

6.2 Definition (Value-Passing Processes)
The set $\mathcal{P}^{\mathrm{VP}}$ is defined by the same grammar as the set $\mathcal{P}$ with the following exceptions.

- Actions μ are now interpreted as in Notation 6.1.
- The list of arguments of a process constant $\mathsf{A}\langle\, \vec{m} \,\rangle$ may contain variables, values, and names.
- A process may also be $[e]P$ for some expression e and a continuation process P: the intution is as in "`if` $\mathcal{E}(e)$ `then` P".

6.3 Remark
Note that $a.P$ is *not* a valid term of $\mathcal{P}^{\mathrm{VP}}$.

6.4 Definition (Free and Bound Names and Variables)
The sets $\mathrm{fn}(P)$, $\mathrm{bn}(P)$, $\mathrm{fv}(P)$, and $\mathrm{bv}(P)$ are defined analogously to the definitions of $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ for the concurrent process expressions.

$$\mathrm{fn}(\mu) \stackrel{\mathrm{def}}{=} \begin{cases} \{\, a \,\} & \text{if } \mu \in \{\, \overline{a}\langle x\rangle, \overline{a}\langle v\rangle, a(x) \,\} \\ \emptyset & \text{if } \mu = \tau \end{cases}$$

$$\mathrm{bn}(\mu) \stackrel{\mathrm{def}}{=} \emptyset$$

$$\mathrm{bv}(\mu) \stackrel{\mathrm{def}}{=} \begin{cases} \emptyset & \text{if } \mu \in \{\, \overline{a}\langle x\rangle, \overline{a}\langle v\rangle, \tau \,\} \\ \{\, x \,\} & \text{if } \mu = a(x) \end{cases}$$

$$\mathrm{fv}(\mu) \stackrel{\mathrm{def}}{=} \begin{cases} \emptyset & \text{if } \mu \in \{\, \overline{a}\langle v\rangle, a(x), \tau \,\} \\ \{\, x \,\} & \text{if } \mu = \overline{a}\langle x\rangle \end{cases}$$

As δ comprises just fewer cases than μ in §6.1, these four definitions also apply to δ.

The notion of α-conversion now also includes the consistent renaming of (input) variables. Substitution, as well as free/bound names/variables of labels, are defined accordingly. However, values may never substitute names or vice versa and substitution also affects expressions.

6.5 Definition (Operational Semantics (Early))
The LTS $(\mathcal{P}^{\mathrm{VP}}, \mathcal{T}^{\mathrm{E}})$ over $\mathcal{L}$ has $\mathcal{P}^{\mathrm{VP}}$ as states, and its transitions $\mathcal{T}^{\mathrm{E}}$ are generated by the following rules:

$$\textsc{sum}_1\colon \frac{M_1 \xrightarrow{\delta} M_1'}{M_1+M_2 \xrightarrow{\delta} M_1'} \qquad \textsc{sum}_2\colon \frac{M_2 \xrightarrow{\delta} M_2'}{M_1+M_2 \xrightarrow{\delta} M_2'}$$

$$\textsc{def}\colon \frac{\{\vec{c}/\vec{a}\}M' \xrightarrow{\delta} P'}{\mathsf{A}\langle\vec{c}\rangle \xrightarrow{\delta} P'}\ \Phi$$

$$\Phi \stackrel{\text{def}}{=} \mathsf{A}(\vec{a}):=M \text{ AND } M=_\alpha M' \text{ AND } \mathrm{cf}(\{\vec{c}/\vec{a}\}, M')$$

$$\textsc{par}_1\colon \frac{P_1 \xrightarrow{\delta} P_1'}{P_1|P_2 \xrightarrow{\delta} P_1'|P_2} \qquad \textsc{par}_2\colon \frac{P_2 \xrightarrow{\delta} P_2'}{P_1|P_2 \xrightarrow{\delta} P_1|P_2'}$$

$$\textsc{res}\colon \frac{P \xrightarrow{\delta} P'}{(\nu a)\,P \xrightarrow{\delta} (\nu a)\,P'}\ a \notin \mathrm{fn}(\delta) \qquad \textsc{tau}\colon \frac{}{\tau.P \xrightarrow{\tau} P}$$

$$\textsc{cond}\colon \frac{P \xrightarrow{\delta} P'}{[e]P \xrightarrow{\delta} P'}\ \mathcal{E}(e) = t \qquad \textsc{out}\colon \frac{}{\overline{a}\langle v\rangle.P \xrightarrow{\overline{a}\langle v\rangle} P}$$

$$\textsc{inp}\colon \frac{}{a(x).P \xrightarrow{av} \{v/x\}P}\ v \in \mathcal{V}$$

$$\textsc{com}_1\colon \frac{P \xrightarrow{\overline{a}\langle v\rangle} P' \qquad Q \xrightarrow{av} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\textsc{com}_2\colon \frac{P \xrightarrow{av} P' \qquad Q \xrightarrow{\overline{a}\langle v\rangle} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

6.6 Definition (Operational Semantics (Late))
The LTS $(\mathcal{P}^{\mathrm{VP}}, \mathcal{T}^{\mathrm{L}})$ over $\mathcal{A}$ has $\mathcal{P}^{\mathrm{VP}}$ as states, and its transitions $\mathcal{T}^{\mathrm{L}}$ are generated by the following rules:

$$\textsc{sum}_1\colon \frac{M_1 \xrightarrow{\mu} M_1'}{M_1+M_2 \xrightarrow{\mu} M_1'} \qquad \textsc{sum}_2\colon \frac{M_2 \xrightarrow{\mu} M_2'}{M_1+M_2 \xrightarrow{\mu} M_2'}$$

$$\textsc{def}\colon \frac{\{\vec{c}/\vec{a}\}M' \xrightarrow{\mu} P'}{\mathsf{A}\langle \vec{c} \rangle \xrightarrow{\mu} P'}\ \Phi$$

$$\Phi \stackrel{\text{def}}{=} \mathsf{A}(\vec{a}) := M \text{ AND } M =_\alpha M' \text{ AND } \mathrm{cf}(\{\vec{c}/\vec{a}\}, M')$$

$$\textsc{par}_1\colon \frac{P_1 \xrightarrow{\mu} P_1'}{P_1|P_2 \xrightarrow{\mu} P_1'|P_2} \qquad \textsc{par}_2\colon \frac{P_2 \xrightarrow{\mu} P_2'}{P_1|P_2 \xrightarrow{\mu} P_1|P_2'}$$

$$\textsc{res}\colon \frac{P \xrightarrow{\mu} P'}{(\nu a)\,P \xrightarrow{\mu} (\nu a)\,P'}\ a \notin \mathrm{fn}(\mu) \qquad \textsc{tau}\colon \frac{}{\tau.P \xrightarrow{\tau} P}$$

$$\textsc{cond}\colon \frac{P \xrightarrow{\mu} P'}{[e]P \xrightarrow{\mu} P'}\ \mathcal{E}(e) = t \qquad \textsc{out}\colon \frac{}{\overline{a}\langle v \rangle.P \xrightarrow{\overline{a}\langle v \rangle} P}$$

$$\textsc{inp}\colon \frac{}{a(x).P \xrightarrow{a(x)} P}$$

$$\textsc{com}_1\colon \frac{P \xrightarrow{\overline{a}\langle v \rangle} P' \qquad Q \xrightarrow{a(x)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid \{v/x\}Q'}$$

$$\textsc{com}_2\colon \frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}\langle v \rangle} Q'}{P \mid Q \xrightarrow{\tau} \{v/x\}P' \mid Q'}$$

6.7 Definition (Early Strong (Bi-)Simulation)
Let $S \subseteq \mathcal{P}^{\mathrm{VP}} \times \mathcal{P}^{\mathrm{VP}}$ be a binary relation.
S is an *early strong simulation* on $(\mathcal{P}^{\mathrm{VP}}, \mathcal{T}^{\mathrm{L}})$ if,
whenever $P \; S \; Q$,

- if $P \xrightarrow{a(x)} P'$ then
  $\boxed{\text{for all } v \in \mathcal{V}}$
  $\boxed{\text{there is } Q'}$ such that
  - $Q \xrightarrow{a(y)} Q'$ and
  - $\{{}^{v}/_{x}\}P' \; S \; \{{}^{v}/_{y}\}Q'$
- if $P \xrightarrow{\mu} P'$ $(\mu \in \{\, \tau \,\} \cup \{\, \overline{a}\langle v \rangle \mid a \in \mathcal{N} \wedge v \in \mathcal{V} \,\})$ then
  there is Q' such that
  - $Q \xrightarrow{\mu} Q'$ and
  - $P' \; S \; Q'$

A binary relation $B \subseteq \mathcal{P}^{\mathrm{VP}} \times \mathcal{P}^{\mathrm{VP}}$ is an early strong bisimulation,
iff, B and B^{-1} are early strong simulations.
The largest early strong bisimulation is denoted $\sim^{\mathrm{E}}$.

6.8 Definition (Late Strong (Bi-)Simulation)
Let $S \subseteq \mathcal{P}^{\mathrm{VP}} \times \mathcal{P}^{\mathrm{VP}}$ be a binary relation.
S is a *late strong simulation* on $(\mathcal{P}^{\mathrm{VP}}, \mathcal{T}^{\mathrm{L}})$ if,
whenever $P \; S \; Q$,

- if $P \xrightarrow{a(x)} P'$ then
  $\boxed{\text{there is } Q'}$ such that
  - $Q \xrightarrow{a(y)} Q'$ and
  - $\boxed{\text{for all } v \in \mathcal{V}} \; . \; \{{}^{v}/_{x}\}P' \; S \; \{{}^{v}/_{y}\}Q'$
- if $P \xrightarrow{\mu} P'$ $(\mu \in \{\, \tau \,\} \cup \{\, \overline{a}\langle v \rangle \mid a \in \mathcal{N} \wedge v \in \mathcal{V} \,\})$ then
  there is Q' such that
  - $Q \xrightarrow{\mu} Q'$ and
  - $P' \; S \; Q'$

A binary relation $B \subseteq \mathcal{P}^{\mathrm{VP}} \times \mathcal{P}^{\mathrm{VP}}$ is a late strong bisimulation,
iff, B and B^{-1} are late strong simulations.
The largest late strong bisimulation is denoted $\sim^{\mathrm{L}}$.

6.9 Definition (Translational Semantics)
We consider a restricted class of $\mathcal{P}^{\mathrm{VP}}$ terms: Let $\mathcal{V} = \{ v_0, \ldots, v_n \}$ be finite and process terms contain neither values nor variables in parameter lists of applications of process constants.
We give an encoding: $[\![\ ]\!] \ : \ \mathcal{P}^{\mathrm{VP}} \to \mathcal{P}$

$$
\begin{array}{rcl}
[\![\tau.P]\!] & \stackrel{\mathrm{def}}{=} & \tau.[\![P]\!] \\
[\![\overline{a}\langle v \rangle.P]\!] & \stackrel{\mathrm{def}}{=} & \overline{a_v}.[\![P]\!] \\
[\![a(x).P]\!] & \stackrel{\mathrm{def}}{=} & \sum_{v \in \mathcal{V}} a_v.[\![\{^v/_x\} P]\!] \\
[\![[e]P]\!] & \stackrel{\mathrm{def}}{=} & \begin{cases} [\![P]\!] & \text{if } \mathcal{E}(e) = \mathsf{t} \\ \mathbf{0} & \text{if } \mathcal{E}(e) = \mathsf{f} \end{cases} \\
\hline
[\![\mathbf{0}]\!] & \stackrel{\mathrm{def}}{=} & \mathbf{0} \\
[\![M_1 + M_2]\!] & \stackrel{\mathrm{def}}{=} & [\![M_1]\!] + [\![M_2]\!] \\
[\![P_1 \mid P_2]\!] & \stackrel{\mathrm{def}}{=} & [\![P_1]\!] \mid [\![P_2]\!] \\
[\![(\boldsymbol{\nu} a)\, P]\!] & \stackrel{\mathrm{def}}{=} & (\boldsymbol{\nu} [\![a]\!])\, [\![P]\!] \\
\hline
[\![\mathsf{A}\langle \vec{a} \rangle]\!] & \stackrel{\mathrm{def}}{=} & \mathsf{A}\langle [\![\vec{a}]\!] \rangle \\
[\![\mathsf{A}(\vec{x}) := M]\!] & \stackrel{\mathrm{def}}{=} & \mathsf{A}([\![\vec{x}]\!]) := [\![M]\!] \\
\hline
[\![\vec{a}]\!] & \stackrel{\mathrm{def}}{=} & \begin{cases} \lambda & \text{if } \vec{a} = \lambda \\ b_{v_0}, \ldots, b_{v_n}, [\![\vec{c}]\!] & \text{if } \vec{a} = b\vec{c} \end{cases} \\
\hline\hline
[\![\overline{a}\langle v \rangle]\!] & \stackrel{\mathrm{def}}{=} & \overline{a_v} \\
[\![av]\!] & \stackrel{\mathrm{def}}{=} & a_v \\
[\![\tau]\!] & \stackrel{\mathrm{def}}{=} & \tau
\end{array}
$$

6.10 Proposition
Let $P, P' \in \mathcal{P}^{\mathrm{VP}}$ with the restrictions from definition 6.9. Then:

$$P \xrightarrow{\mu}_{\mathcal{T}^{\mathrm{E}}} P' \text{ iff } [\![P]\!] \xrightarrow{[\![\mu]\!]} [\![P']\!].$$

6.11 Corollary
The translational semantics of definition 6.9 is not sound in the sense of proposition 6.10 if the *late* semantics and *late* strong bisimulation is used on $\mathcal{P}^{\mathrm{VP}}$ instead of their early counterparts.

6.12 Proposition
For value passing CCS we have:

$$\Delta \subset =_\alpha \subset \equiv \subset \sim^{\mathrm{L}} \subset \sim^{\mathrm{E}} = \sim^{\mathcal{T}^{\mathrm{E}}} \subset \nabla$$

Where $\sim^{\mathcal{T}^{\mathrm{E}}}$ is strong bisimilarity using the early semantics $\mathcal{T}^{\mathrm{E}}$.

6.13 Notation (Polyadic Communication)
The *polyadic actions* $\overline{a}\langle\vec{v}\rangle$ and $a(\vec{x})$
(with $\vec{x}$ pairwise different)
transmit many values at a time.
All definitions are straightforwardly generalized.

6.14 Remark
Note that $a.P$ *is* a valid term of the polyadic variant of $\mathcal{P}^{\mathrm{VP}}$.

6.15 Proposition
(Value-Passing) CCS is Turing-powerful.

6.16 Proposition
The halting problem for Turing machines can be reduced to the existence of infinite sequences of internal transitions.

7 Name-Passing and Reaction Semantics

7.1 Notation

We use the following sets of entities with corresponding meta-variables:

$$\begin{array}{lll} \mathcal{N} & \textit{names} & a, b, c, \ldots, x, y, z \\ \mathcal{A} & \textit{actions} & \mu \ ::= \ \overline{x}\langle y \rangle \mid x(y) \mid \tau \end{array}$$

7.2 Definition (Mobile Processes)

The set $\mathcal{P}^\pi$ of *π-calculus process expressions* and the set $\mathcal{M}^\pi$ of *π-calculus summations* are defined by the following syntax:

$$\begin{array}{rcl} P & ::= & M \mid P|P \mid (\nu a)\, P \mid !\, P \\ M & ::= & \mathbf{0} \mid \mu.P \mid M + M \end{array}$$

7.3 Definition (Free and Bound Names)

The sets $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ are defined as for concurrent process expressions, based on its definition on actions, and adapted in the case of prefixes.

$$\mathrm{fn}(\mu) \stackrel{\text{def}}{=} \begin{cases} \{ y \} & \text{if } \mu = y(x) \\ \{ y, z \} & \text{if } \mu = \overline{y}\langle z \rangle \\ \emptyset & \text{if } \mu = \tau \end{cases}$$

$$\mathrm{bn}(\mu) \stackrel{\text{def}}{=} \begin{cases} \{ x \} & \text{if } \mu = y(x) \\ \emptyset & \text{if } \mu \in \{ \overline{y}\langle z \rangle, \tau \} \end{cases}$$

$$\mathrm{fn}(\mu.P) \stackrel{\text{def}}{=} \mathrm{fn}(\mu) \cup (\mathrm{fn}(P) \setminus \mathrm{bn}(\mu))$$

$$\mathrm{bn}(\mu.P) \stackrel{\text{def}}{=} \mathrm{bn}(\mu) \cup \mathrm{bn}(P)$$

7.4 Definition (Process Contexts)
A *π-calculus process context* $C[\cdot]$ is defined by the following syntax:

$$\begin{aligned} C[\cdot] \;::=\; & [\cdot] \mid \mu.C[\cdot] + M \mid M + \mu.C[\cdot] \\ & \mid P|C[\cdot] \mid C[\cdot]|P \mid (\boldsymbol{\nu} a)\, C[\cdot] \mid \mathbf{!}\, C[\cdot] \end{aligned}$$

The *elementary process contexts* are

$$\begin{array}{ccc} (\boldsymbol{\nu} a)\,[\cdot] & \mu.[\cdot] + M & M + \mu.[\cdot] \\ \mathbf{!}\,[\cdot] & [\cdot] \mid P & P \mid [\cdot] \end{array}$$

7.5 Definition (Process Congruence)
Let $\cong$ be an equivalence relation over $\mathcal{P}^\pi$.
Then $\cong$ is said to be a process congruence,
if for *all* contexts $C[\cdot]$, $P \cong Q$ implies $C[P] \cong C[Q]$.

7.6 Proposition
Let $\cong$ be an equivalence relation over $\mathcal{P}^\pi$.
Then $\cong$ is a process congruence, if and only if
it is preserved by all elementary contexts;
i.e., $P \cong Q$ implies all of the following:

$$\begin{array}{cc} \mu.P + M \cong \mu.Q + M & P|R \cong Q|R \\ M + \mu.P \cong M + \mu.Q & R|P \cong R|Q \\ \mathbf{!}\, P \cong \mathbf{!}\, Q & (\boldsymbol{\nu} a)\, P \cong (\boldsymbol{\nu} a)\, Q \end{array}$$

7.7 Definition (Structural congruence)
Structural congruence, written $\equiv$, is the (smallest) process congruence over $\mathcal{P}^\pi$ determined by the following equations.

1. $=_\alpha$ (now for two binding operators!)
2. commutative monoid laws for $(\mathcal{M}^\pi, +, \mathbf{0})$
3. commutative monoid laws for $(\mathcal{P}^\pi, \mid, \mathbf{0})$
4. rules for restriction:

$$\begin{aligned}(\boldsymbol{\nu} a)\,(P \mid Q) &\equiv P \mid (\boldsymbol{\nu} a)\,Q, \quad \text{if } a \notin \mathrm{fn}(P)\\ (\boldsymbol{\nu} ab)\,P &\equiv (\boldsymbol{\nu} ba)\,P\\ (\boldsymbol{\nu} a)\,\mathbf{0} &\equiv \mathbf{0}\end{aligned}$$

5. rule for replication: $\mathbf{!}\,P \equiv P|\mathbf{!}\,P$

7.8 Definition (Standard Form)
A π-calculus process expression

$$(\boldsymbol{\nu}\vec{a})\,(\ M_1|\ldots|M_m|\mathbf{!}\,Q_1|\ldots|\mathbf{!}\,Q_n\ )$$

where each M_i is a non-empty sum, is said to be in *standard form*, if each Q_j is itself in standard form.
If $m = 0$ then $M_1|\ldots|M_m$ means $\mathbf{0}$.
If $n = 0$ then $\mathbf{!}\,Q_1|\ldots|\mathbf{!}\,Q_n$ means $\mathbf{0}$.
If $\vec{a}$ is empty then there is no restriction.

7.9 Theorem
Every π-calculus process term is structurally congruent to some standard form.

7.10 Definition

The reaction relation $\mapsto$ over $\mathcal{P}^{\pi}$ is generated by the following rules:

$$\textsc{tau:}\ \frac{}{\tau.P+M \mapsto P}$$

$$\textsc{react:}\ \frac{}{\overline{y}\langle z\rangle.P+M \mid y(x).Q+N \mapsto P \mid \{^{z}/_{x}\}Q}$$

$$\textsc{par:}\ \frac{P \mapsto P'}{P|Q \mapsto P'|Q} \qquad \textsc{res:}\ \frac{P \mapsto P'}{(\nu a)\,P \mapsto (\nu a)\,P'}$$

$$\textsc{struct:}\ \frac{P \mapsto P'}{Q \mapsto Q'}\ P \equiv Q \text{ and } P' \equiv Q'$$

There are many variants of the π-calculus available in the literature. Here, we introduce two widely-used ones.

7.11 Notation (Polyadism)
We use the following sets of entities and meta-variables:

$$\begin{array}{lll} \mathcal{N} & \textit{names} & a, b, c, \ldots, x, y, z \\ \mathcal{A} & \textit{actions} & \mu \ ::= \ \tau \mid y(\vec{x}) \mid \overline{y}\langle\vec{z}\rangle \end{array}$$

where the x_i in $(\vec{x})$ are pairwise distinct, and
where in *bound output* $(\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle$, we require $\{\,\vec{x}\,\} \subseteq \{\,\vec{z}\,\}$.
We write $\overline{y}\langle\vec{z}\rangle$ for $(\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle$ when $\vec{x}$ is empty.
Then, the definitions of 7 are just re-applied, except for the adaptation of the rule

$$\textsc{react:}\ \frac{}{\overline{y}\langle\vec{z}\rangle.P + M \mid y(\vec{x}).Q + N \mapsto P \mid \{^{\vec{z}}/_{\vec{x}}\}Q}\ |\vec{z}| = |\vec{x}|$$

7.12 Definition (Asynchrony)
The *asynchronous* π-calculus is the subset of the standard (then called *synchronous*) π-calculus given by:
1. constraining sending to the form $\overline{y}\langle\tilde{z}\rangle$
   (without any suffix);
2. removing the summation operator

The syntax of $\mathcal{P}^{\mathrm{A}}$ is generated by the BNF-grammar:

$$P \ ::= \ \mathbf{0} \mid \overline{y}\langle\tilde{z}\rangle \mid y(\tilde{x}).P \mid P|P \mid (\boldsymbol{\nu}a)\,P \mid \mathbf{!}\,P$$

where terms of the form $\overline{y}\langle\tilde{z}\rangle$ are called *messages*.
Then, many definitions of 7 are re-applied, with an the adaptation of the rule

$$\textsc{react:}\ \frac{}{\overline{y}\langle\vec{z}\rangle \mid y(\vec{x}).Q \mapsto \{^{\vec{z}}/_{\vec{x}}\}Q}\ |\vec{z}| = |\vec{x}|$$

8 Name-Passing and Labeled Semantics

8.1 Notation (Labels)
We use the following:

$$\mathcal{L} \quad \textit{labels} \quad \pi \;::=\; \tau \;\mid\; y(\vec{x}) \;\mid\; y\vec{z} \;\mid\; (\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle$$

where the x_i in $(\vec{x})$ are pairwise distinct, and
where in *bound output* $(\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle$, we require $\{\,\vec{x}\,\} \subseteq \{\,\vec{z}\,\}$.
We write $\overline{y}\langle\vec{z}\rangle$ for $(\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle$ when $\vec{x}$ is empty.
We write $\overline{y}(\boldsymbol{\nu}z)$ for $(\boldsymbol{\nu}z)\,\overline{y}\langle z\rangle$.

8.2 Definition (Free and Bound Names)
The sets $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ are adapted to the case of polyadic actions and labels. Let $\gamma \in \{\,\mu, \pi\,\}$.

$$\mathrm{n}(\gamma) \stackrel{\text{def}}{=} \begin{cases} \{\,y, \vec{x}\,\} & \text{if } \gamma = y(\vec{x}) \\ \{\,y, \vec{z}\,\} & \text{if } \gamma \in \{\,y\vec{z}, \overline{y}\langle\vec{z}\rangle, (\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle\,\} \\ \emptyset & \text{if } \gamma = \tau \end{cases}$$

$$\mathrm{bn}(\gamma) \stackrel{\text{def}}{=} \begin{cases} \{\,\vec{x}\,\} & \text{if } \gamma \in \{\,y(\vec{x}), (\boldsymbol{\nu}\vec{x})\,\overline{y}\langle\vec{z}\rangle\,\} \\ \emptyset & \text{if } \gamma \in \{\,y\vec{z}, \overline{y}\langle\vec{z}\rangle, \tau\,\} \end{cases}$$

$$\mathrm{fn}(\gamma) \stackrel{\text{def}}{=} \mathrm{n}(\gamma) \setminus \mathrm{bn}(\gamma)$$

8.3 Remark
Let $\gamma \in \{\,\mu, \pi\,\}$.
In the sequel, the phrase "$\mathrm{bn}(\alpha)$ fresh" means that $\mathrm{bn}(\alpha)$ is disjoint with the free names of all other mentioned processes.

8.4 Definition (Operational Semantics)
The LTS $(\mathcal{P}^\pi, \mathcal{T})$ of π-calculus expressions over $\mathcal{L}$ has $\mathcal{P}^\pi$ as states, and its transitions $\mathcal{T}$ are generated by:

$$\textsc{tau:}\ \frac{}{\tau.P \xrightarrow{\ \tau\ } P} \qquad \textsc{out:}\ \frac{}{\overline{a}\langle\vec{b}\rangle.P \xrightarrow{\ \overline{a}\langle\vec{b}\rangle\ } P}$$

$$\textsc{inp:}\ \frac{}{a(\vec{x}).P \xrightarrow{\ a\vec{b}\ } \{{}^{\vec{b}}/_{\vec{x}}\}P}\ \{\vec{b}\} \subseteq \mathcal{N}$$

$$\textsc{res:}\ \frac{P \xrightarrow{\ \pi\ } P'}{(\boldsymbol{\nu} c)\, P \xrightarrow{\ \pi\ } (\boldsymbol{\nu} c)\, P'}\ c \notin \mathrm{n}(\pi)$$

$$\textsc{open:}\ \frac{P \xrightarrow{\ (\boldsymbol{\nu}\vec{b})\,\overline{a}\langle\vec{z}\rangle\ } P'}{(\boldsymbol{\nu} c)\, P \xrightarrow{\ (\boldsymbol{\nu} c\vec{b})\,\overline{a}\langle\vec{z}\rangle\ } P'}\ \{\vec{z}\} \ni c \notin \{a\} \cup \{\vec{b}\}$$

$$\textsc{par}_1\textsc{:}\ \frac{P_1 \xrightarrow{\ \pi\ } P_1'}{P_1 \mid P_2 \xrightarrow{\ \pi\ } P_1' \mid P_2}\ \mathrm{bn}(\pi) \cap \mathrm{fn}(P_2) = \emptyset$$

$$\textsc{close}_1\textsc{:}\ \frac{P_1 \xrightarrow{\ a\vec{b}\ } P_1' \qquad P_2 \xrightarrow{\ (\boldsymbol{\nu}\vec{c})\,\overline{a}\langle\vec{b}\rangle\ } P_2'}{P_1 \mid P_2 \xrightarrow{\ \tau\ } (\boldsymbol{\nu}\vec{c})\,(P_1' \mid P_2')}\ \{\vec{c}\} \cap \mathrm{fn}(P_1) = \emptyset$$

$$\textsc{sum}_1\textsc{:}\ \frac{P_1 \xrightarrow{\ \pi\ } P_1'}{P_1 + P_2 \xrightarrow{\ \pi\ } P_1'} \qquad \textsc{rep:}\ \frac{P \mid \mathbf{!}\,P \xrightarrow{\ \pi\ } P'}{\mathbf{!}\,P \xrightarrow{\ \pi\ } P'}$$

$$\textsc{alpha:}\ \frac{Q \xrightarrow{\ \pi\ } Q'}{P \xrightarrow{\ \pi\ } Q'}\ P =_\alpha Q$$

where the obvious symmetric counterparts for the rules ($\textsc{par}_1$), ($\textsc{close}_1$) and ($\textsc{sum}_1$) are omitted.

8.5 Definition (Late Operational Semantics)
The LTS $(\mathcal{P}^\pi, \mathcal{T}^{\mathrm{L}})$ of sequential process expressions over $\mathcal{A}$ has $\mathcal{P}^\pi$ as states, and its transitions $\mathcal{T}^{\mathrm{L}}$ are generated by the following rules:

$$\textsc{pre:}\ \frac{}{\mu.P \xrightarrow{\ \mu\ } P}$$

$$\textsc{res:}\ \frac{P \xrightarrow{\ \pi\ } P'}{(\nu c)\,P \xrightarrow{\ \pi\ } (\nu c)\,P'}\ c \notin \mathrm{n}(\pi)$$

$$\textsc{open:}\ \frac{P \xrightarrow{\ (\nu \vec{b})\,\overline{a}\langle \vec{z}\rangle\ } P'}{(\nu c)\,P \xrightarrow{\ (\nu c\vec{b})\,\overline{a}\langle \vec{z}\rangle\ } P'}\ \{\,\vec{z}\,\} \ni c \notin \{\,a\,\} \cup \{\,\vec{b}\,\}$$

$$\textsc{par}_1\textsc{:}\ \frac{P_1 \xrightarrow{\ \pi\ } P_1'}{P_1 \mid P_2 \xrightarrow{\ \pi\ } P_1' \mid P_2}\ \mathrm{bn}(\pi) \cap \mathrm{fn}(P_2) = \emptyset$$

$$\textsc{close}_1\textsc{:}\ \frac{P_1 \xrightarrow{\ a(\vec{x})\ } P_1' \qquad P_2 \xrightarrow{\ (\nu \vec{c})\,\overline{a}\langle \vec{b}\rangle\ } P_2'}{P_1 \mid P_2 \xrightarrow{\ \tau\ } (\nu \vec{c})\,(\{\vec{b}/\vec{x}\}P_1' \mid P_2')}\ \{\,\vec{c}\,\} \cap \mathrm{fn}(P_1) = \emptyset$$

$$\textsc{sum}_1\textsc{:}\ \frac{P_1 \xrightarrow{\ \pi\ } P_1'}{P_1 + P_2 \xrightarrow{\ \pi\ } P_1'} \qquad \textsc{rep:}\ \frac{P \mid \mathbf{!}\,P \xrightarrow{\ \pi\ } P'}{\mathbf{!}\,P \xrightarrow{\ \pi\ } P'}$$

$$\textsc{alpha:}\ \frac{Q \xrightarrow{\ \pi\ } Q'}{P \xrightarrow{\ \pi\ } Q'}\ P =_\alpha Q$$

where the obvious symmetric counterparts for the rules ($\textsc{par}_1$), ($\textsc{close}_1$) and ($\textsc{sum}_1$) are omitted.

For simplicity, the following two definitions are only shown for the monadic case.

8.6 Definition (Early Strong (Bi-)Simulation)
Let $S \subseteq \mathcal{P}^\pi \times \mathcal{P}^\pi$ be a binary relation.
S is an *early strong simulation* on $(\mathcal{P}^\pi, \mathcal{T}^{\mathrm{L}})$ if,
whenever $P\ S\ Q$,

- if $P \xrightarrow{a(x)} P'$ then
  $\boxed{\text{for all } u \in \mathcal{N}}\ \boxed{\text{there is } Q'}$ such that
  - $Q \xrightarrow{a(x)} Q'$ and
  - $\{{}^{u}/_{x}\}P'\ S\ \{{}^{u}/_{x}\}Q'$
- if $P \xrightarrow{\mu} P'$ $(\mu \in \{\tau\} \cup \{\overline{a}\langle v\rangle, \overline{a}(\boldsymbol{\nu} v) \mid a \in \mathcal{N} \wedge v \in \mathcal{N}\})$ and $\mathrm{bn}(\mu)$ fresh then
  there is Q' such that
  - $Q \xrightarrow{\mu} Q'$ and
  - $P'\ S\ Q'$

A binary relation $B \subseteq \mathcal{P}^\pi \times \mathcal{P}^\pi$ is an early strong bisimulation, iff, B and B^{-1} are early strong simulations.
The largest early strong bisimulation is denoted $\sim^{\mathrm{E}}$.

8.7 Definition (Late Strong (Bi-)Simulation)
Let $S \subseteq \mathcal{P}^\pi \times \mathcal{P}^\pi$ be a binary relation.
S is an *late strong simulation* on $(\mathcal{P}^\pi, \mathcal{T}^{\mathrm{L}})$ if,
whenever $P\ S\ Q$,

- if $P \xrightarrow{a(x)} P'$ then
  $\boxed{\text{there is } Q'}$ such that
  - $Q \xrightarrow{a(x)} Q'$ and
  - $\boxed{\text{for all } u \in \mathcal{N}}\ \{{}^{u}/_{x}\}P'\ S\ \{{}^{u}/_{x}\}Q'$
- if $P \xrightarrow{\mu} P'$ $(\mu \in \{\tau\} \cup \{\overline{a}\langle v\rangle, \overline{a}(\boldsymbol{\nu} v) \mid a \in \mathcal{N} \wedge v \in \mathcal{N}\})$ and $\mathrm{bn}(\mu)$ fresh then
  there is Q' such that
  - $Q \xrightarrow{\mu} Q'$ and
  - $P'\ S\ Q'$

A binary relation $B \subseteq \mathcal{P}^\pi \times \mathcal{P}^\pi$ is a late strong bisimulation, iff, B and B^{-1} are late strong simulations.
The largest late strong bisimulation is denoted $\sim^{\mathrm{L}}$.

9 Congruence

9.1 Notation
$R^{=}$ denotes the symmetric closure $R \cup R^{-1}$ of the relation R.

9.2 Definition (Distinction)
1. A *distinction* $D \subseteq (\mathcal{N} \times \mathcal{N})$ is a
   finite symmetric irreflexive relation on names.
2. Let $N \subseteq \mathcal{N}$. Then

$$N^{\neq} \stackrel{\text{def}}{=} \{\, (x,y) \in N \times N \mid x \neq y \,\}$$

   is a distinction.
3. A substitution σ *respects* a distinction D
   if $(x,y) \in D$ implies $\sigma x \neq \sigma y$.
4. A *D-congruence* is an equivalence that is preserved by those contexts that do not use the names in D as "hole-binding" names.

9.3 Remark
Note that if a substitution σ *respects* a distinction D, then σD is also a distinction.

9.4 Definition (Strong Open Bisimulation)
The set $\{ \sim^D \mid D \text{ is a distinction} \}$ on $(\mathcal{P}^\pi, \mathcal{T})$ is the largest family of symmetric relations such that if $P \sim^D Q$ and σ respects D, then

- if $\sigma P \xrightarrow{\pi} P'$ with $\mathrm{bn}(\pi)$ fresh (i.e., with $\mathrm{bn}(\pi) \cap \mathrm{fn}(\sigma P | \sigma Q) = \emptyset$), then there is Q' with $\sigma Q \xrightarrow{\pi} Q'$ and $P' \sim^{\sigma D \cup D'} Q'$, where[1]

$$D' \stackrel{\text{def}}{=} \begin{cases} \{\mathrm{bn}(\pi)\}^{\neq} \cup \big(\mathrm{bn}(\pi) \times \mathrm{fn}(\sigma P | \sigma Q)\big)^{=} & \\ & \mathrm{bn}(\pi) \neq \emptyset \\ \emptyset & \mathrm{bn}(\pi) = \emptyset \end{cases}$$

The weak version ($\approx^D$) is defined as usual by allowing for a weak simulating transition in each case.

9.5 Theorem
$\sim^D$ and $\approx^D$ are *D-congruences*.

[1] Beware of the standard non-distinguishing treatment of bound input when using a late transition semantics instead of the early transiton semantics that is used here.