Parallel and Distributed Systems

Summary

Student： 陳鵬 (Chen Peng)
StudentID： 6930-30-2948
Lab： Yoshikawa & Ma Laboratory
E-mail: chenpeng.acmer@yahoo.com
Date: Jan.30 2019

Chapter 1 Process

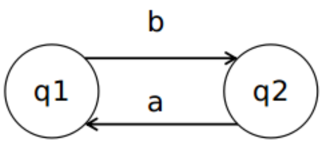
1.1 Automata and Process

- **Definition:**

  A finite-state automata can be represented as (Q, q_o, T) where

  1. Q is a finite set of states
  2. $q_0 \in Q$, whereq_0 is an initial state
  3. $T \subseteq Q \times Act \times Q$, representing transitions
  4. Act is a finite set of actions.
  5. $(q, a, q') \in \mathcal{T}$ can be written as $q \xrightarrow{a} q'$.

- **Example:**



  There are two transitions in the example:

$$q1 \xrightarrow{b} q2$$
$$q2 \xrightarrow{a} q1$$

1.2 Trace

- **Definition:**

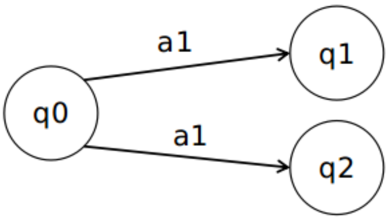
  The trace of a deterministic finite-state automata (Q, q_0, T, Act) is defined as follows:

$$\left\{a_1, a_2, a_3, ... | q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3 ...\right\}$$

  In this case, a state q and an action a can uniquely determine a transition. $q \xrightarrow{a} q'$

  Nondeterministic algorithm:

  In computer science, a nondeterministic algorithm is an algorithm that, even for the same input, can exhibit different behaviors on different runs, as opposed to a deterministic algorithm.



In a nondeterministic finite-state automata, a state can exhibit different behaviours on one action.

1.3 Process

- **Definition:**
  In general, a process is a series of actions or steps taken in order to achieve a particular end. Here, a process means a computing entity that can communicate with outer world.
  Notation:

$$N \text{ is a set of names.}$$
$$\overline{N} := \{\overline{a} | a \in N\} \text{ is defined after } N\text{, which is the set of co-names.}$$
$$\text{Here, } \overline{\overline{a}} = a$$
$$L := N \cup \overline{N} \text{ represents the set of labels.}$$

- **Process expression in BNF grammar:**
  Notice: BNF here is a series of rewriting rules.

$$\begin{aligned} A_1 < a_1 \ldots a_{n_1} > &= P_1 \\ A_2 < b_1 \ldots b_{n_2} > &= P_2 \end{aligned}$$
$$\text{where, } a_1 \ldots b_{a_1} \text{ and } b_1 \ldots b_{n_2} \text{ are parameters of P1 and P2.}$$

$$\begin{aligned} P := A &< a_1, \ldots, a_n > \\ &| a_1.P_1 + \cdots + a_n.P_n \\ &| \varnothing \text{ (do-nothing)} \end{aligned}$$

$$\text{where } a_1.P_1 + \cdots + a_n.P_n \text{ means the process will do}$$
$$\begin{array}{ll} a_1.P_1 & \text{or} \\ a_2.P_2 & \text{or} \\ a_3.P_3 & \text{or} \\ \ldots\ldots & \end{array}$$

1.4 Structural Congruence

- **Basics:**

  The relation $\equiv$ is the smallest relation:

$$a_1P_1 + \ldots a_nP_n \equiv a_{q_1}P_{q_1} + \cdots + a_{q_n}P_{q_n}\ (q_1, \ldots, q_n \text{ is an reordering of } 1, \ldots, n)$$
$$A < a_1 \ldots a_n > \equiv [a_1, \ldots, a_n / b_1, \ldots, b_n]\, P$$
$$\text{where } a_1, \ldots, a_n \text{ is replaced with } b_1, \ldots, b_n$$
$$P_1 \equiv Q_1, \ldots, P_n \equiv Q_n \text{ imply } a_1P_1 + \cdots + a_nP_n \equiv a_1Q_1 + \cdots + a_nQ_n$$

Reflexicity, symmetricity and transitivity:

$$P \equiv P \qquad \text{(reflexicity)}$$
$$P \equiv Q \text{ implies } Q \equiv P \qquad \text{(symmetricity)}$$
$$P \equiv Q \text{ and } Q \equiv R \text{ imply } P \equiv R \qquad \text{(transitivity)}$$

Inference rules:

(The equation(s) below can be inferred by upper equation(s).)

$$\frac{}{P \equiv P}$$
$$\frac{P\equiv Q}{Q\equiv P}$$
$$\frac{P\equiv Q, Q\equiv R}{P\equiv R}$$
$$\frac{P_1\equiv Q_1,\ldots,P_n\equiv Q_n}{a_1P_1+\cdots+a_nP_n\equiv a_1Q_1+\cdots+a_nQ_n}$$

1.5 Labelled Transition Systems(LTS) for process

- **Definition:**

  A labelled transition system is a tuple (S, A, $\longrightarrow$) where S is a set of states, A is a set of labels and is a set of labelled transitions (i.e., a subset of S $\times$ A $\times$ S). (p, α,q) $\in \longrightarrow$ is written as:

$$p \xrightarrow{\alpha} q$$

  which means p can do action a, and evolves to q.

- **Example:**

  Suppose there is a buffer keeping 0 or 1. In the beginning, the buffer has no value.

$$N = \{in_0, in_1, out_0, out_1\}$$

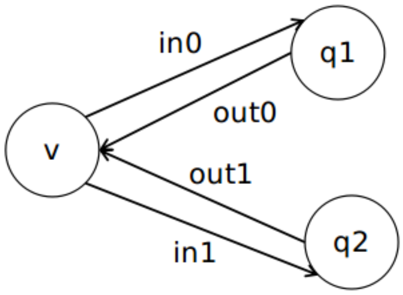
  - If the buffer has no value:
    1. do action in_0, the buffer keeps 0.
    2. do action in_1, the buffer keeps 1.
  - If the buffer has a value v:
    1. if $v = 0$, do action $\overline{out_0}$, erase the value.
    2. if $v = 1$, do action $\overline{out_1}$, erase the value.

$$Buff_\epsilon = in_0.Buff_0 + in_1.Buff_1$$

$$Buff_0 = \overline{out_0}.Buff_\epsilon$$

$$Buff_1 = \overline{out_1}.Buff_\epsilon$$

Diagram of the buff:



1.6 Strong simulation for process

- **Definition:**

  A binary relation $\mathcal{S} \subseteq \mathbb{P}_{roc} \times \mathbb{P}_{roc}$ is called a strong simulation iff, in which, $\mathbb{P}_{\mathrm{roc}}$ is set of processes.

  $\forall P, Q \in \mathbb{P}_{\mathrm{roc}}$ and $\forall a \in Act$
  If $P\mathcal{S}Q$ (also means $(P, Q) \in \mathcal{S}$) and $P \xrightarrow{a} P'$, then

  $\exists Q' \in \mathbb{P}_{\mathrm{roc}}, Q \xrightarrow{a} Q'$ and $P'\mathcal{S}Q'$. $P'\mathcal{S}Q'$ also means Q can simulate P.

  The following diagram shows a strong simulaion S, (Q simulates P, that is $P \preceq Q$):

$$\begin{array}{ccc} P & \mathcal{S} & Q \\ \downarrow^{a} & & \downarrow^{a} \\ P' & \mathcal{S} & \exists Q' \end{array}$$

1.7 Strong bisimulation for process

- **Definition:**

  A binary relation $\mathcal{S} \subseteq \mathbb{P}_{\mathrm{roc}} \times \mathbb{P}_{\mathrm{roc}}$ is said to be a strong bisimulation if both of $\mathcal{S}$ and $\mathcal{S}^{-1}$ are strong simulation. ($\mathcal{S}^{-1}$ means $\{(P_2, P_1) \mid (P_1, P_2) \in \mathcal{S}\}$)

  If there exists a strong bisimulation between P and Q, P is said to be strong bisimulation to Q, and written $P \sim Q$.

$$P_1 \preceq P_2 \Longleftrightarrow \exists \mathcal{S}.P_1\mathcal{S}\mathcal{P}_\in \wedge \mathcal{S} \text{ is a strong simulation.}$$

To prove $P \sim Q$:

1. Given a binary relation $\mathcal{S}$, $s.t.(P,Q) \in \mathcal{S}$
2. Show that $\mathcal{S}$ is a strong simulation.
3. $\mathcal{S}^{-1}$ is a strong simulation.

- **Equivalence:**

The relation $\sim$ is an equivalence relation.

1. $\forall P \quad P \sim P$
2. $\forall P, Q \quad P \sim Q \to Q \sim P$
2. $\forall P, Q, R \quad P \sim Q \wedge Q \sim R \to P \sim R$

Proof of $\forall P \quad P \sim P$:

Obviously, a process is able to simulate itself.

$$\begin{array}{ccc} P & \mathcal{S} & P \\ \downarrow^a & & \downarrow^a \\ P' & \mathcal{S} & \exists P' \end{array}$$

Proof of $\forall P, Q \quad P \sim Q \to Q \sim P$:

Because $P \sim Q$, there exists a strong bisimulation S between them:

$$\begin{array}{ccc} P & \mathcal{S} & Q \\ \downarrow^a & & \downarrow^a \\ P' & \mathcal{S} & \exists Q' \end{array}$$

And we have:

$$\begin{array}{ccc} Q & \mathcal{S}^{-1} = R & P \\ \downarrow^a & & \downarrow^a \\ Q' & \mathcal{S}^{-1} = R & \exists P' \\ & R^{-1} = S & \end{array}$$

Obviously, there must exist a strong bisimulation R between Q and P.

Proof of $\forall P, Q, R \quad P \sim Q \wedge Q \sim R \to P \sim R$:

For any action executed by P, Q can always simulate it, and so does R. (Because R can simulate Q). Therefore, R can simulate P.

$$\begin{array}{ccccc} P & \mathcal{S}1 & Q & \mathcal{S}2 & R \\ \downarrow^a & & \downarrow^a & & \downarrow^a \\ P' & \mathcal{S}1 & \exists Q' & \mathcal{S}2 & \exists R' \end{array}$$

Similarly, we can show that P can simulate R.

$$\begin{array}{ccccc} P & \mathcal{S}1 & Q & \mathcal{S}2 & R \\ \downarrow^a & & \downarrow^a & & \downarrow^a \\ \exists P' & \mathcal{S}1 & \exists Q' & \mathcal{S}2 & R' \end{array}$$

Strengbeweis for $\forall P, Q, R \quad P \sim Q \land Q \sim R \to P \sim R$**:**

First, choose P, Q, and R arbitrarily. Suppose $P \sim Q$ and $Q \sim R$.

There are $\mathcal{S}_1$ and $\mathcal{S}_2$.

1. $\mathcal{S}_1$ and $\mathcal{S}_2$ are strong simulations.
2. S_1^{-1} and S_2^{-1} are strong simulations.
3. $(P, Q) \in \mathcal{S}_1$
4. $(Q, R) \in \mathcal{S}_2$

Then, set $T := \mathcal{S}_1\mathcal{S}_2$, which means a composition of $\mathcal{S}_1$ and $\mathcal{S}_2 := \{(P', R') \mid \exists Q' \quad s.t. \quad (P', Q') \in \mathcal{S}_1 \quad \text{and } (Q', R') \in \mathcal{S}_2\}$, then $(P, R) \in T$. $(P\mathcal{S}_1\mathcal{S}_2R \iff \exists Q. \quad P\mathcal{S}_1Q \quad \text{and} \quad Q\mathcal{S}_2R)$

Here, T is a strong simulation.

Proof:

There are two lemmas:

1. If $\mathcal{S}_1$ and $\mathcal{S}_2$ are strong simulations. $\mathcal{S}_1\mathcal{S}_2$ is also a strong simulation.
2. $(\mathcal{S}_1\mathcal{S}_2)^{-1} = \mathcal{S}_2^{-1}\mathcal{S}_1^{-1}$

Because both $\mathcal{S}_1$ and $\mathcal{S}_2$ are strong simulationa. Thus, $T = \mathcal{S}_1\mathcal{S}_2$ is a strong simulation.

And, T^{-1} is also a strong simulation.

Proof:

$T^{-1} = (\mathcal{S}_1\mathcal{S}_2)^{-1} = \mathcal{S}_2^{-1}\mathcal{S}_1^{-1}$ both are strong simulations.

Therefore, $P \sim R$.

Chapter 2 Calculus of Communicating Systems

Introduced by Robin Milner, 1980.

CCS is a process calculus, whose actions model indivisible communications between exactly two participants, providing description of process networks with static topologies.

2.1 CCS Basics (channels, actions, process)

- **Channels & Labels:**

Let $A = \{a, b, c, ...\}$ be a set of **channel names**.
$L = A \cup \overline{A}$ is a set of **labels** where $\overline{A} = \{\overline{x} | x \in A\}$
Here, elements of $\overline{A}$ are called **co-names**.
$\overline{a}$ is the co-name of a. Obviously, $\overline{\overline{a}} = a$

- **Actions:**

$Act = L \cup \{\tau\}$ is a set of actions where
τ is the **internal** action.
input: $a.P \xrightarrow{a} P$
output: $\overline{a}.P \xrightarrow{\overline{a}} P$

- **CCS Process:** The set of CCS processes is defined by BNF grammer:

$P ::= A < a_1, a_2, ..., a_n >$	process identifier
$\mid \ \sum a_i P_i$	$prefixing$
$\mid \ a.P1$	P performs action a, and continue as process P1
$\mid \ P1 \mid P2$	parallel composition
$\mid \ P1 + P2$	proceed either as P1 or P2
$\mid \ P1[b/a]$	relabelling: P1 with all actions a renamed as b
$\mid$ new a in P	scope (of hide) of a
$\mid \ P1 \backslash a$	restriction: P1 without action a
$\mid \ \varnothing$	$inaction$

Quick examples:

$$\varnothing$$

$$A < a >, A < a, b >, B < a, b, c > ...$$

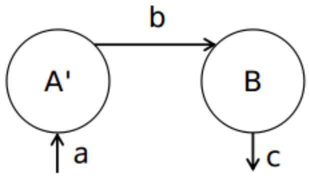
$$a.\varnothing + \overline{b}A < a >$$

$R + a.P|b.Q\backslash L \quad \underline{\underline{def}} \quad R + ((a.P)|(b.(Q\backslash L)))$. ($\underline{\underline{def}}$ means equation)

- **Intuition of each construct:**

1. Communication happens if sending/receiving actions are concurrently executed.

$$A' \stackrel{def}{=} \overline{b}aA'$$
$$B \stackrel{def}{=} b\overline{c}B$$



Besides, $A'|B \to aA'|\overline{c}B$

2. Non-deterministic choice is much like sequence process.

$$\begin{array}{ccc} (ac+bd)|(\overline{a}\overline{c}+\overline{b}\overline{d}) & & \\ \downarrow & & \downarrow \\ c|\overline{c} & & d|\overline{d} \\ \downarrow & & \downarrow \\ 0|0 & = 0 = & 0|0 \end{array}$$

3. New a in P hides actions in P related to the name a. (overlap)

$a|$ new a in $\overline{a}.0$ will not become 0

Because a is now local to $\overline{a}.0$ that cannot communicate with.

4. Action τ can be executed without communication.

$$\begin{array}{cc} \tau.a0 + \tau.b0 & \\ \downarrow & \downarrow \\ a0 & b0 \end{array}$$

2.2 Formal semantics of CSS

- **Bound names, free names:**

We write $f_n(P)$ for the following set:

$$f_n(0) := \emptyset$$
$$f_n\left(P_1|P_2\right) := f_n\left(P_1\right) \vee f_n\left(P_2\right)$$
$$f_n\left(A < a_1, \ldots, a_n >\right) = \{a_1, \ldots, a_n\}$$
$$f_n(\text{ new } a \text{ in } P) = f_n(P)\backslash\{a\}$$
$$f_n\left(\sum_{i\in I} a_i.P_i\right) = \cup_{i\in I} f_n\left(P_i\right) \vee \text{ name } \left(a_i\right)$$

$$\text{where } name(a) = a$$
$$name(\overline{a}) = a$$
$$name(\tau) = \varnothing$$

2.3 Labelled Transition Systems(LTS) for CSS

Basic model for representing reactive, concurrent, parallel, communicating systems.

- **Definition:**

$$< P, P_0, L, T >$$
$$P = \text{set of states}$$
$$P_0 = \text{an initial state}$$
$$L = \text{set of labels (e.g. communication actions, etc)}$$
$$T \subseteq S \times L \times S = \text{set of transitions}$$
$$P_i \xrightarrow{a} P_i' \text{ where } (P_i, a, P_i') \in T$$

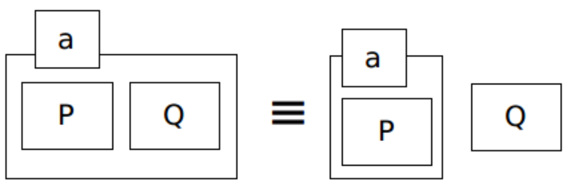
- **Def.(structural congruence):**

Part1: Basics

$$P|0 \equiv P$$
$$P|Q \equiv Q|P$$
$$P|(Q|R) \equiv (P|Q)|R$$
$$P_1 |(P_2| (P_3|P_4)) \equiv (P_1|P_2)| (P_3|P_4) \equiv P_1 |P_2| P_3 |P_4$$

Part2: Scoping or hiding

$$\text{new } a \text{ in } (P|Q) \equiv (\text{ new } a \text{ in } P) \big| Q \text{ if } a \notin f_n(Q)$$
$$\text{new } a \text{ in } 0 \equiv 0$$
$$\text{new } a \text{ in } b \text{ in } P \equiv \text{ new } b \text{ in new } a \text{ in } P$$
$$\text{(We sometimes write new } a, b \text{ in } P \text{ for new } a \text{ in } b \text{ in } P)$$



Part3: Renaming

$$\text{If } P \equiv Q \text{ then } P|R \equiv Q|R$$
$$\text{and new } a \text{ in } P \equiv \text{ new } a \text{ in } Q$$
$$\text{and } a.P \equiv a.Q$$

Part4: Reflexivity, symmetricity and transitivity

$$P \equiv P$$
$$P \equiv Q \text{ implies } Q \equiv P$$
$$P \equiv Q \text{ and } Q \equiv P \text{ implies } P \equiv R$$

- **Def.**($P_i \xrightarrow{a} P_i'$)**:**

  Summation:

  $$\sum_{i \in I} \alpha_i . P_i \xrightarrow{\alpha_i} P_i$$
  E.g.
  $$a.0 + b.0 \xrightarrow{a} 0$$
  $$a.0 + b.0 \xrightarrow{b} 0$$

  React:

  $$\frac{P \xrightarrow{\lambda} P' \qquad Q \xrightarrow{\bar{\lambda}} Q'}{P \Big| Q \xrightarrow{\tau} P' \Big| Q'}$$
  where $\lambda ::= a|\overline{a}$ and λ cannot be τ
  $$\text{Besides, } \overline{\lambda} = \begin{cases} \overline{a} & \text{if } \lambda = a \\ a & \text{if } \lambda = \overline{a} \end{cases}$$

  LPar and RPar:

  $$\frac{P \xrightarrow{\alpha} P'}{P \Big| Q \xrightarrow{\alpha} P' \Big| Q}$$
  $$\frac{Q \xrightarrow{\alpha} Q'}{P \Big| Q \xrightarrow{\alpha} P \Big| Q'}$$
  where $\alpha := a|\overline{a}|\tau$

  Res.:

  $$\frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin \{a, \overline{a}\}}{\text{new } a \text{ in } P \xrightarrow{\alpha} \text{new } a \text{ in } P'}$$

  Structural Cong.:

  $$\frac{P \equiv P' \qquad P' \xrightarrow{\alpha} Q' \qquad Q' \equiv Q}{P \xrightarrow{\alpha} Q}$$

2.4 Strong Bisimulation for for CSS

- **Def.(bisimulation for CCS):**

  $$\mathcal{R} \subseteq \mathbb{P}roc \times \mathbb{P}roc \text{ is a strong simulation, if}$$
  $$\forall P, Q, P', \alpha. \quad (P, Q) \in \mathcal{R} \text{ and } P \xrightarrow{\alpha} P'$$

then

$$\exists Q'. \quad Q \xrightarrow{\alpha} Q' \text{ and } (P', Q') \in \mathcal{R}$$

$$\boxed{\begin{array}{ccc} P & \mathcal{R} & Q \\ \downarrow^{\alpha} & & \Downarrow^{\alpha} \\ P' & \mathcal{R} & \exists Q' \end{array}}$$

$\mathcal{R}$ is called a strong bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are both strong simulations. If there is a strong bisimulation between P and Q, then we write $P \sim Q$.

Example: Semaphores

- **Def.(strong simulation/bisimulation up to ≡):**

$\mathcal{R}$ is a strong simulation up to ≡, if $\forall P, Q, P', \alpha. \quad (P, Q) \in \mathcal{R}$ and $P \xrightarrow{\alpha} P'$ imply $\exists Q'.P' \equiv \mathcal{R} \equiv Q'$
$\mathcal{R}$ is a strong bisimulation up to ≡ if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are both strong simulations.
Besides, if ∃ is a strong bisimulation up to ≡,

$$(P, Q) \in \mathcal{R}$$
$$\text{then } P \sim Q$$

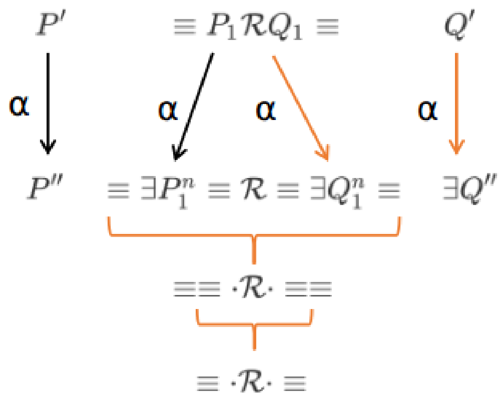
$$\boxed{\begin{array}{ccc} P & \mathcal{R} & Q \\ \downarrow^{\alpha} & & \downarrow^{\alpha} \\ P' & \equiv \mathcal{R} \equiv & \exists Q' \end{array}}$$

Proof:
Suppose $\mathcal{R}$ is a strong bisimulation up to ≡ and $(P, Q) \in \mathcal{R}$. We show that $\equiv \mathcal{R} \equiv$ is a strong bisimulation and $(P, Q) \in \equiv \mathcal{R} \equiv$, which is obvious because ≡ is reflexive: $P \equiv P\mathcal{R}Q \equiv Q$
There are two lemmas:

1. ≡ is a strong bisimulation.
2. If $P \equiv\equiv Q$, then $P \equiv Q$.

In order to prove $\equiv R \equiv$ is a strong bisimulation, assume that $(P', Q') \in \equiv \mathcal{R} \equiv$ and $P' \rightarrow^{\alpha} P''$

The following process shows the transformation:



And we can also prove that $(\equiv \mathcal{R} \equiv)^{-1}$ is a strong simulation. (Omitted)

Chapter 3 The π Calculus

The π-calculus a process calculus. It allows channel names to be communicated along the channels themselves, and in this way it is able to describe concurrent computations whose network configuration may change during the computation.
The π-calculus is simple, and has very few terms and so is a very small language, yet is very expressive.

3.1 Basics

- **Definition:**

$\pi ::= x(y)$	receive value y through x
$\mid \quad \overline{x} < y >$	send value y through x
$\mid \quad \tau$	internal action(s)
$P ::= \sum_i \pi_i P_i$	non-deterministic choice
$\mid \quad P1 \mid P2$	parallel composition
$\mid$ new a in P	generate a new name a
$\mid \quad !P$	replication of P

- **Example:**

$Server\ :=!S(x)$

(receiving a channel from a client)

new a in P

(generate a new channel named a)

$\overline{x} < a > .a(y)$

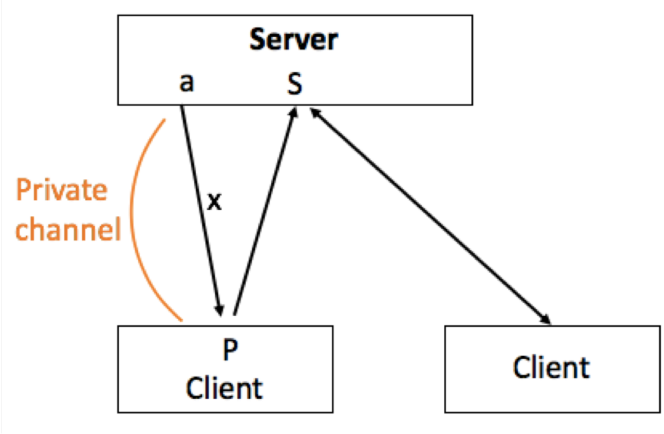
(send back the new channel, and use it for communication)

$Client\ :=$ new P in

(generate a new name P)

$\overline{S} < P > .P(x).\overline{x} < 1 >$

(send P to the server S, wait for the reply to P, and communicate using x)

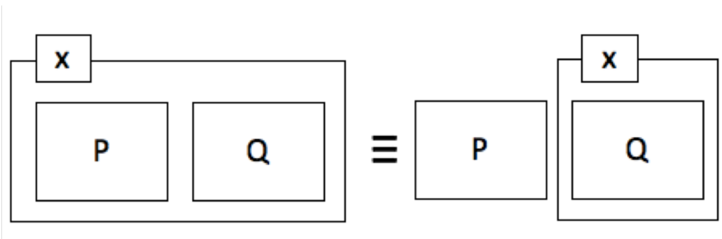


3.2 Structural Congruence

- **Definition:**

  Part1:

$$P|0 \equiv P \qquad P|Q \equiv Q|P \qquad P|(Q|R) \equiv (P|Q)|R$$
$$\text{new } x \text{ in } (P|Q) \equiv P|\text{ new } x \text{ in } Q \qquad x \notin f_n(P)$$



$$\text{new } x \text{ in } 0 \equiv 0$$
$$\text{new } x \text{ in new } y \text{ in } P \equiv \text{new } y \text{ in new } x \text{ in } P$$
$$\text{Besids, } !P \equiv P|!P \equiv P|P|!P \ldots.$$
$$\text{If P and Q differs only in bound names, then } P \equiv Q$$
$$\begin{cases} \text{new } x \text{ in } \overline{x} < y > \equiv \text{ new } z \text{ in } \overline{z} < y > \\ x(y) \cdot \overline{y} < z > \equiv x(a) \cdot \overline{a} < z > \\ \quad \text{(Receive a value from x and name it y, then send z via y.)} \\ \quad \text{(Receive a value from x and name it a, then send z via a.)} \end{cases}$$

Part2:

$$P \equiv Q \text{ implies}$$
$$\begin{cases} \text{new } x \text{ in } P \equiv \text{ new } x \text{ in } Q \\ P|R \equiv Q|R \\ !P \equiv !Q \end{cases}$$
$$P_i \equiv Q_i \text{ imply } \sum \pi_i P_i \equiv \sum \pi_i Q_i$$

Part3:

$$P \equiv P \qquad \text{(reflexicity)}$$
$$P \equiv Q \text{ implies } Q \equiv P \qquad \text{(symmetricity)}$$
$$P \equiv Q \text{ and } Q \equiv R \text{ imply } P \equiv R \qquad \text{(transitivity)}$$

3.3 Reduction Semantics

- **Definition:**

  Part1:

$$\tau P + \sum \pi_i Q_i \rightarrow P$$

$$\left(x(y)P + \sum \pi_i R_i\right) \Big| \left(\overline{x} < z > Q + \sum \pi_i' R_i'\right) \rightarrow [z/y]P \Big| Q \quad \text{(React)}$$

Process obtained by replacing y in P with z.

  Part2:

$$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \quad \text{(Par)}$$

$$\frac{P \rightarrow P'}{\text{new } x \text{ in } P \rightarrow \text{ new } x \text{ in } P'} \quad \text{(Res)}$$

$$\frac{P \equiv P' \qquad P' \rightarrow Q' \qquad Q' \equiv Q}{P \rightarrow Q} \quad \text{(Struct)}$$

  Part3:

$$P \xrightarrow{\overline{a}} Q$$

$$\frac{P \xrightarrow{a} Q \qquad P' \xrightarrow{\overline{a}} Q'}{P \Big| P' \xrightarrow{\tau} Q \Big| Q'}$$

- **Example:**

$$Server := !S(x)$$

$$\text{new a in P}$$

$$\overline{x} < a > .a(y)$$

$$Client := \text{new P in}$$

$$\overline{S} < P > .P(x).\overline{x} < n >$$

$$\begin{aligned}
Server|Client &\equiv Server|S(x)\text{new a in}\overline{x} < a > .a(y)|Client \\
&\equiv Server|\text{new P in}[\text{S(x), new a in}\overline{x} < a > .a(y)|\overline{S} < P > .P(x).\overline{x} < n >] \\
&\rightarrow Server|\text{new P in}[\text{new a in}\overline{P} < a > .a(y)|P(x).\overline{x} < n >] \\
&\equiv Server|\text{new P in, new a in}[\overline{P} < a > .a(y)|P(x).\overline{x} < n >] \\
&\rightarrow Server|\text{new P in, new a in}[a(y)|\overline{a} < n >] \\
&\rightarrow Server|\text{new P in, new a in}[0|0] \\
&\equiv Server|0 \\
&\equiv Server
\end{aligned}$$