

Process Algebra

Luís Soares Barbosa



Universidade do Minho



UNITED NATIONS
UNIVERSITY

UNU-EGOV

Architecture & Calculi Course Unit

Universidade do Minho

States & Actions

What's in an action?

- elementary unit of behaviour that can **execute itself atomically in time** (no duration), after which it terminates successfully
- is a **latency for interaction**

... and a process?

- ... an abstract representation of a system's **behaviour**
- a term specifying a labelled transition systems

A language for processes

CCS: a prototypical process algebra

- *Calculus of Communicating Systems* [Milner, 1980]

- Actions:

$$Act ::= a \mid \bar{a} \mid \tau$$

for $a \in N$, N denoting a set of **names**

- Processes:
 - No sequential composition: but **action prefix** $a.$
 - No distinction between **termination** and **deadlock** (why?)
 - Communication by **binary handshake**
(of complementary actions)

Examples

Buffers

1-position buffer: $A(in, out) \hat{=} in.\overline{out}.0$

... non terminating: $B(in, out) \hat{=} in.\overline{out}.B$

... with two output ports: $C(in, o_1, o_2) \hat{=} in.(\overline{o_1}.C + \overline{o_2}.C)$

... non deterministic: $D(in, o_1, o_2) \hat{=} in.\overline{o_1}.D + in.\overline{o_2}.D$

... with parameters: $B(in, out) \hat{=} in(x).\overline{out}\langle x \rangle.B$

Examples

n -position buffers

1-position buffer:

$$S \hat{=} (B\langle in, m \rangle \mid B\langle m, out \rangle) \setminus \{m\}$$

n -position buffer:

$$B_n \hat{=} (B\langle in, m_1 \rangle \mid B\langle m_1, m_2 \rangle \mid \cdots \mid B\langle m_{n-1}, out \rangle) \setminus \{m_i \mid i < n\}$$

Examples

mutual exclusion

$$Sem \hat{=} get.put.Sem$$

$$P_i \hat{=} \overline{get}.c_i.\overline{put}.P_i$$

$$S \hat{=} (Sem \mid (\prod_{i \in I} P_i)) \setminus_{\{get, put\}}$$

CCS Syntax

The set \mathbb{P} of **processes** is the set of all terms generated by the following BNF:

$$E ::= A(x_1, \dots, x_n) \mid a.E \mid \sum_{i \in I} E_i \mid E_0 \mid E_1 \mid E \setminus K$$

for $a \in Act$ and $K \subseteq L$

Abbreviations

$$E_0 + E_1 \stackrel{abv}{=} \sum_{i \in \{0,1\}} E_i$$

$$0 \stackrel{abv}{=} \sum_{i \in \emptyset} E_i$$

CCS Syntax

The set \mathbb{P} of **processes** is the set of all terms generated by the following BNF:

$$E ::= A(x_1, \dots, x_n) \mid a.E \mid \sum_{i \in I} E_i \mid E_0 \mid E_1 \mid E \setminus K$$

for $a \in Act$ and $K \subseteq L$

Abbreviations

$$E_0 + E_1 \stackrel{\text{abv}}{=} \sum_{i \in \{0,1\}} E_i$$

$$\mathbf{0} \stackrel{\text{abv}}{=} \sum_{i \in \emptyset} E_i$$

CCS Syntax

Process declaration

$$A(\vec{x}) \hat{=} E_A$$

with $fn(E_A) \subseteq \vec{x}$ (where $fn(P)$ is the set of **free** variables of P).

- used as, e.g., $A(a, b, c) \hat{=} a.b.0 + c.A\langle d, e, f \rangle$

Process declaration: fixed point expression

$$\underline{fix} (X = E_X)$$

- syntactic **substitution** over \mathbb{P} , cf.
 - $\{c/b\} a.b.0$

CCS Syntax

Process declaration

$$A(\vec{x}) \hat{=} E_A$$

with $fn(E_A) \subseteq \vec{x}$ (where $fn(P)$ is the set of **free** variables of P).

- used as, e.g., $A(a, b, c) \hat{=} a.b.0 + c.A\langle d, e, f \rangle$

Process declaration: fixed point expression

$$\underline{fix} (X = E_X)$$

- syntactic **substitution** over \mathbb{P} , cf.
 - $\{c/b\} a.b.0$

Semantics

Two-level semantics

- **arquitectural**, expresses a notion of **similar assembly configurations** and is expressed through a **structural congruence** relation;
- **behavioural** given by **transition rules** which express how system's components interact

Semantics

Structural congruence

\equiv over \mathbb{P} is given by the closure of the following conditions:

- for all $A(\vec{x}) \hat{=} E_A$, $A(\vec{y}) \equiv \{\vec{y}/\vec{x}\} E_A$,
(i.e., **folding/unfolding** preserve \equiv)
- α -conversion (i.e., replacement of bounded variables).
- both $|$ and $+$ originate, with $\mathbf{0}$, **Abelian monoids**
- for all $a \notin fn(P)$ $(P | Q) \setminus \{a\} \equiv P | Q \setminus \{a\}$
- $\mathbf{0} \setminus \{a\} \equiv \mathbf{0}$

Semantics

$$\frac{}{a.p \longrightarrow p} \text{ (prefix)}$$

$$\frac{\{\vec{k}/\vec{x}\} p_A \xrightarrow{a} p'}{A(\vec{k}) \xrightarrow{a} p'} \text{ (ident) (if } A(\vec{x}) \hat{=} p_A)$$

$$\frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \text{ (sum - l)}$$

$$\frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'} \text{ (sum - r)}$$

Semantics

$$\frac{p \xrightarrow{a} p'}{p \mid q \xrightarrow{a} p' \mid q} \text{ (par - l)} \qquad \frac{q \xrightarrow{a} q'}{p \mid q \xrightarrow{a} p \mid q'} \text{ (par - r)}$$

$$\frac{p \xrightarrow{a} p' \quad q \xrightarrow{\bar{a}} q'}{p \mid q \xrightarrow{\tau} p' \mid q'} \text{ (react)}$$

$$\frac{p \xrightarrow{a} p'}{p \setminus \{k\} \xrightarrow{a} p' \setminus \{k\}} \text{ (res)} \quad (\text{if } a \notin \{k, \bar{k}\})$$

Compatibility

Lemma

Structural congruence preserves transitions:

if $p \xrightarrow{a} p'$ and $p \equiv q$ there exists a process q' such that $q \xrightarrow{a} q'$ and $p' \equiv q'$.

Semantics

These rules define a **LTS**

$$\{\xrightarrow{a} \subseteq \mathbb{P} \times \mathbb{P} \mid a \in Act\}$$

Relation \xrightarrow{a} is defined **inductively** over process structure entailing a semantic description which is

Structural *i.e.*, each process **shape** (defined by the most external combinator) has a type of transitions

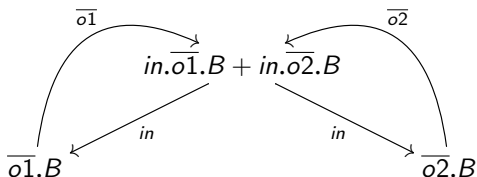
Modular *i.e.*, a process transition is defined from transitions in its sup-processes

Complete *i.e.*, all possible transitions are inferred from these rules

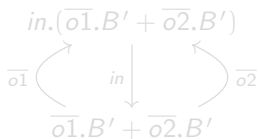
static vs dynamic combinators

Examples

$$B \hat{=} in.\overline{o1}.B + in.\overline{o2}.B$$

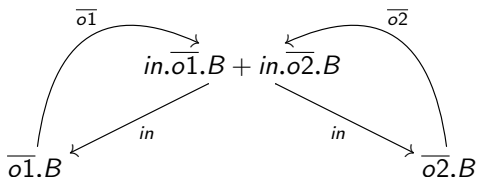


compare with $B' \hat{=} in.(\overline{o1}.B' + \overline{o2}.B')$

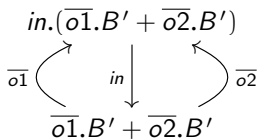


Examples

$$B \hat{=} in.\overline{o1}.B + in.\overline{o2}.B$$



compare with $B' \hat{=} in.(\overline{o1}.B' + \overline{o2}.B')$



Data parameters

Language \mathbb{P} is extended to \mathbb{P}_V over a data universe V , a set V_e of expressions over V and a evaluation $Val : V_e \rightarrow V$

Example

$$B \hat{=} in(x).B'_x$$

$$B'_v \hat{=} \overline{out}\langle v \rangle.B$$

- Two prefix forms: $a(x).E$ and $\bar{a}\langle e \rangle.E$ (actions as ports)
- Data parameters: $A_S(x_1, \dots, x_n) \hat{=} E_A$, with $S \in V$ and each $x_i \in L$
- Conditional combinator: *if b then P , if b then P_1 else P_2*

Clearly

$$if\ b\ then\ P_1\ else\ P_2 \stackrel{abv}{=} (if\ b\ then\ P_1) + (if\ \neg b\ then\ P_2)$$

Data parameters

Language \mathbb{P} is extended to \mathbb{P}_V over a data universe V , a set V_e of expressions over V and a evaluation $Val : V_e \rightarrow V$

Example

$$B \hat{=} in(x).B'_x$$

$$B'_v \hat{=} \overline{out}\langle v \rangle.B$$

- Two prefix forms: $a(x).E$ and $\bar{a}\langle e \rangle.E$ (**actions** as **ports**)
- Data parameters: $A_S(x_1, \dots, x_n) \hat{=} E_A$, with $S \in V$ and each $x_i \in L$
- Conditional combinator: *if b then P , if b then P_1 else P_2*

Clearly

$$if\ b\ then\ P_1\ else\ P_2 \stackrel{abv}{=} (if\ b\ then\ P_1) + (if\ \neg b\ then\ P_2)$$

Data parameters

Additional semantic rules

$$\frac{}{a(x).E \xrightarrow{a(v)} \{v/x\}E} \text{ (prefix}_i\text{)} \quad \text{for } v \in V$$

$$\frac{}{\bar{a}\langle e \rangle.E \xrightarrow{\bar{a}\langle v \rangle} E} \text{ (prefix}_o\text{)} \quad \text{for } \text{Val}(e) = v$$

$$\frac{E_1 \xrightarrow{a} E'}{\text{if } b \text{ then } E_1 \text{ else } E_2 \xrightarrow{a} E'} \text{ (if}_1\text{)} \quad \text{for } \text{Val}(b) = \text{true}$$

$$\frac{E_2 \xrightarrow{a} E'}{\text{if } b \text{ then } E_1 \text{ else } E_2 \xrightarrow{a} E'} \text{ (if}_2\text{)} \quad \text{for } \text{Val}(b) = \text{false}$$

Back to \mathbb{P}

Encoding in the basic language: $T(\cdot) : \mathbb{P}_V \longrightarrow \mathbb{P}$

$$T(a(x).E) = \sum_{v \in V} a_v \cdot T(\{v/x\}E)$$

$$T(\bar{a}\langle e \rangle.E) = \bar{a}_e \cdot T(E)$$

$$T\left(\sum_{i \in I} E_i\right) = \sum_{i \in I} T(E_i)$$

$$T(E \mid F) = T(E) \mid T(F)$$

$$T(E \setminus K) = T(E) \setminus_{\{a_v \mid a \in K, v \in V\}}$$

and

$$T(\text{if } b \text{ then } E) = \begin{cases} T(E) & \text{if } \text{Val}(b) = \text{true} \\ \mathbf{0} & \text{if } \text{Val}(b) = \text{false} \end{cases}$$

An example

$$P \hat{=} (E_1 \mid E_2 \mid \dots \mid E_n) \setminus_K$$

The chance machine

$$IO \hat{=} m.\overline{bank}.(lost.\overline{loss}.IO + rel(x).\overline{win}\langle x \rangle.IO)$$

$$B_n \hat{=} bank.\overline{max}\langle n+1 \rangle.left(x).B_x$$

$$Dc \hat{=} max(z).(\overline{lost}.left\langle z \rangle.Dc + \sum_{1 \leq x \leq z} \overline{rel}\langle x \rangle.\overline{left}\langle z-x \rangle.Dc)$$

$$M_n \hat{=} (IO \mid B_n \mid Dc) \setminus_{\{bank, max, left, lost, rel\}}$$

Processes are 'prototypical' transition systems

... hence all definitions apply:

$E \sim F$

- Processes E, F are **bisimilar** if there exist a bisimulation S st $\{\langle E, F \rangle\} \in S$.
- A binary relation S in \mathbb{P} is a **(strict) bisimulation** iff, whenever $(E, F) \in S$ and $a \in Act$,

$$\text{i) } E \xrightarrow{a} E' \Rightarrow F \xrightarrow{a} F' \wedge (E', F') \in S$$

$$\text{ii) } F \xrightarrow{a} F' \Rightarrow E \xrightarrow{a} E' \wedge (E', F') \in S$$

i.e.,

$$\sim = \bigcup \{S \subseteq \mathbb{P} \times \mathbb{P} \mid S \text{ is a (strict) bisimulation}\}$$

Processes are 'prototypical' transition systems

Example: $S \sim M$

$$T \hat{=} i.\bar{k}.T$$

$$R \hat{=} k.j.R$$

$$S \hat{=} (T \mid R) \setminus \{k\}$$

$$M \hat{=} i.\tau.N$$

$$N \hat{=} j.i.\tau.N + i.j.\tau.N$$

through **bisimulation**

$$R = \{ \langle S, M \rangle, \langle (\bar{k}.T \mid R) \setminus \{k\}, \tau.N \rangle, \langle (T \mid j.R) \setminus \{k\}, N \rangle, \\ \langle (\bar{k}.T \mid j.R) \setminus \{k\}, j.\tau.N \rangle \}$$

Example: Semaphores

A semaphore

$$Sem \hat{=} get.put.Sem$$

n -semaphores

$$Sem_n \hat{=} Sem_{n,0}$$

$$Sem_{n,0} \hat{=} get.Sem_{n,1}$$

$$Sem_{n,i} \hat{=} get.Sem_{n,i+1} + put.Sem_{n,i-1}$$

(for $0 < i < n$)

$$Sem_{n,n} \hat{=} put.Sem_{n,n-1}$$

Sem_n can also be implemented by the parallel composition of n Sem processes:

$$Sem^n \hat{=} Sem \mid Sem \mid \dots \mid Sem$$

Example: Semaphores

A semaphore

$$Sem \hat{=} get.put.Sem$$

n -semaphores

$$Sem_n \hat{=} Sem_{n,0}$$

$$Sem_{n,0} \hat{=} get.Sem_{n,1}$$

$$Sem_{n,i} \hat{=} get.Sem_{n,i+1} + put.Sem_{n,i-1}$$

(for $0 < i < n$)

$$Sem_{n,n} \hat{=} put.Sem_{n,n-1}$$

Sem_n can also be implemented by the parallel composition of n Sem processes:

$$Sem^n \hat{=} Sem \mid Sem \mid \dots \mid Sem$$

A \sim -calculus

Lemma

$$E \equiv F \Rightarrow E \sim F$$

- **proof idea:** show that $\{(E + E, E) \mid E \in \mathbb{P}\} \cup Id_{\mathbb{P}}$ is a **bisimulation**

Lemma

$$(E \setminus_K) \setminus_{K'} \sim E \setminus_{(K \cup K')}$$

$$E \setminus_K \sim E$$

$$\text{if } \mathbb{L}(E) \cap (K \cup \bar{K}) = \emptyset$$

$$(E \mid F) \setminus_K \sim E \setminus_K \mid F \setminus_K$$

$$\text{if } \mathbb{L}(E) \cap \overline{\mathbb{L}(F)} \cap (K \cup \bar{K}) = \emptyset$$

- **proof idea:** discuss whether S is a **bisimulation**:

$$S = \{(E \setminus_K, E) \mid E \in \mathbb{P} \wedge \mathbb{L}(E) \cap (K \cup \bar{K}) = \emptyset\}$$

A \sim -calculus

Lemma

$$E \equiv F \Rightarrow E \sim F$$

- **proof idea:** show that $\{(E + E, E) \mid E \in \mathbb{P}\} \cup Id_{\mathbb{P}}$ is a **bisimulation**

Lemma

$$(E \setminus_K) \setminus_{K'} \sim E \setminus_{(K \cup K')}$$

$$E \setminus_K \sim E$$

$$\text{if } \mathbb{L}(E) \cap (K \cup \bar{K}) = \emptyset$$

$$(E \mid F) \setminus_K \sim E \setminus_K \mid F \setminus_K$$

$$\text{if } \mathbb{L}(E) \cap \overline{\mathbb{L}(F)} \cap (K \cup \bar{K}) = \emptyset$$

- **proof idea:** discuss whether S is a **bisimulation**:

$$S = \{(E \setminus_K, E) \mid E \in \mathbb{P} \wedge \mathbb{L}(E) \cap (K \cup \bar{K}) = \emptyset\}$$

\sim is a congruence

congruence is the name of **modularity** in Mathematics

- **process combinators** preserve \sim

Lemma

Assume $E \sim F$. Then,

$$a.E \sim a.F$$

$$E + P \sim F + P$$

$$E \mid P \sim F \mid P$$

$$E \setminus \kappa \sim F \setminus \kappa$$

- **recursive definition** preserves \sim

\sim is a congruence

congruence is the name of **modularity** in Mathematics

- **process combinators** preserve \sim

Lemma

Assume $E \sim F$. Then,

$$a.E \sim a.F$$

$$E + P \sim F + P$$

$$E \mid P \sim F \mid P$$

$$E \setminus \kappa \sim F \setminus \kappa$$

- **recursive definition** preserves \sim

\sim is a congruence

- First \sim is extended to **processes with variables**:

$$E \sim F \equiv \forall \tilde{p}. E[\tilde{P}/\tilde{X}] \sim F[\tilde{P}/\tilde{X}]$$

- Then prove:

Lemma

- i) $\tilde{P} \hat{=} \tilde{E} \Rightarrow \tilde{P} \sim \tilde{E}$
 where \tilde{E} is a family of process expressions and \tilde{P} a family of process **identifiers**.
- ii) Let $\tilde{E} \sim \tilde{F}$, where \tilde{E} and \tilde{F} are families of recursive process expressions over a family of process **variables** \tilde{X} , and define:

$$\tilde{A} \hat{=} \tilde{E}[\tilde{A}/\tilde{X}] \quad \text{and} \quad \tilde{B} \hat{=} \tilde{F}[\tilde{B}/\tilde{X}]$$

Then

$$\tilde{A} \sim \tilde{B}$$

The expansion theorem

Every process is equivalent to the sum of its derivatives

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

understood?

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

clear?

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

The expansion theorem

Every process is equivalent to the sum of its derivatives

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

understood?

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

clear?

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

The expansion theorem

Every process is equivalent to the sum of its derivatives

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

understood?

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

clear?

$$E \sim \sum \{a.E' \mid E \xrightarrow{a} E'\}$$

The expansion theorem

The usual definition (based on the **concurrent canonical form**):

$$\begin{aligned}
 E \sim & \sum \{ f_i(a).(E_1[f_1] \mid \dots \mid E_i'[f_i] \mid \dots \mid E_n[f_n]) \setminus_K \mid \\
 & E_i \xrightarrow{a} E_i' \wedge f_i(a) \notin K \cup \overline{K} \} \\
 + & \\
 & \sum \{ \tau.(E_1[f_1] \mid \dots \mid E_i'[f_i] \mid \dots \mid E_j'[f_j] \mid \dots \mid E_n[f_n]) \setminus_K \mid \\
 & E_i \xrightarrow{a} E_i' \wedge E_j \xrightarrow{b} E_j' \wedge f_i(a) = \overline{f_j(b)} \}
 \end{aligned}$$

for $E \hat{=} (E_1[f_1] \mid \dots \mid E_n[f_n]) \setminus_K$, with $n \geq 1$

The expansion theorem

Corollary (for $n = 1$ and $f_1 = id$)

$$(E + F) \setminus_K \sim E \setminus_K + F \setminus_K$$
$$(a.E) \setminus_K \sim \begin{cases} \mathbf{0} & \text{if } a \in (K \cup \bar{K}) \\ a.(E \setminus_K) & \text{otherwise} \end{cases}$$

Example

$$S \sim M$$

$$\begin{aligned}
 S &\sim (T \mid R) \setminus_{\{k\}} \\
 &\sim i.(\bar{k}.T \mid R) \setminus_{\{k\}} \\
 &\sim i.\tau.(T \mid j.R) \setminus_{\{k\}} \\
 &\sim i.\tau.(i.(\bar{k}.T \mid j.R) \setminus_{\{k\}} + j.(T \mid R) \setminus_{\{k\}}) \\
 &\sim i.\tau.(i.j.(\bar{k}.T \mid R) \setminus_{\{k\}} + j.i.(\bar{k}.T \mid R) \setminus_{\{k\}}) \\
 &\sim i.\tau.(i.j.\tau.(T \mid j.R) \setminus_{\{k\}} + j.i.\tau.(T \mid j.R) \setminus_{\{k\}})
 \end{aligned}$$

Let $N' = (T \mid j.R) \setminus_{\{k\}}$.

This expands into $N' \sim i.j.\tau.(T \mid j.R) \setminus_{\{k\}} + j.i.\tau.(T \mid j.R) \setminus_{\{k\}}$,

Therefore $N' \sim N$ and $S \sim i.\tau.N \sim M$

- requires result on **unique** solutions for recursive process equations

Observable transitions

$$\Longrightarrow^a \subseteq \mathbb{P} \times \mathbb{P}$$

- $L \cup \{\epsilon\}$
- A \Longrightarrow^ϵ -transition corresponds to zero or more **non observable** transitions
- inference rules for \Longrightarrow^a :

$$\frac{}{E \Longrightarrow^\epsilon E} (O_1)$$

$$\frac{E \xrightarrow{\tau} E' \quad E' \Longrightarrow^\epsilon F}{E \Longrightarrow^\epsilon F} (O_2)$$

$$\frac{E \Longrightarrow^\epsilon E' \quad E' \xrightarrow{a} F' \quad F' \Longrightarrow^\epsilon F}{E \Longrightarrow^a F} (O_3) \quad \text{for } a \in L$$

Example

$$T_0 \hat{=} j.T_1 + i.T_2$$

$$T_1 \hat{=} i.T_3$$

$$T_2 \hat{=} j.T_3$$

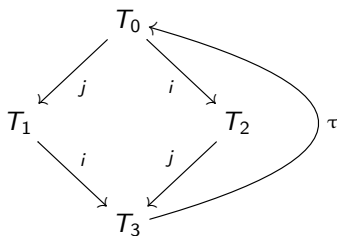
$$T_3 \hat{=} \tau.T_0$$

and

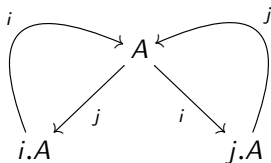
$$A \hat{=} i.j.A + j.i.A$$

Example

From their graphs,



and



we conclude that $T_0 \approx A$ (why?).

Observational equivalence

$$E \approx F$$

- Processes E, F are **observationally equivalent** if there exists a weak bisimulation S st $\{\langle E, F \rangle\} \in S$.
- A binary relation S in \mathbb{P} is a **weak bisimulation** iff, whenever $(E, F) \in S$ and $a \in L \cup \{\epsilon\}$,

$$\text{i) } E \xRightarrow{a} E' \Rightarrow F \xRightarrow{a} F' \wedge (E', F') \in S$$

$$\text{ii) } F \xRightarrow{a} F' \Rightarrow E \xRightarrow{a} E' \wedge (E', F') \in S$$

i.e.,

$$\approx = \bigcup \{S \subseteq \mathbb{P} \times \mathbb{P} \mid S \text{ is a weak bisimulation}\}$$

Observational equivalence

Properties

- **as expected:** \approx is an **equivalence** relation
- **basic property:** for any $E \in \mathbb{P}$,

$$E \approx \tau.E$$

(**proof idea:** $id_{\mathbb{P}} \cup \{(E, \tau.E) \mid E \in \mathbb{P}\}$ is a weak bisimulation)

- **weak vs. strict:**

$$\sim \subseteq \approx$$

Is \approx a congruence?

Lemma

Let $E \approx F$. Then, for any $P \in \mathbb{P}$ and $K \subseteq L$,

$$a.E \approx a.F$$

$$E \mid P \approx F \mid P$$

$$E \setminus_K \approx F \setminus_K$$

but

$$E + P \approx F + P$$

does **not** hold, in general.

Is \approx a congruence?

Lemma

Let $E \approx F$. Then, for any $P \in \mathbb{P}$ and $K \subseteq L$,

$$a.E \approx a.F$$

$$E \mid P \approx F \mid P$$

$$E \setminus_K \approx F \setminus_K$$

but

$$E + P \approx F + P$$

does **not** hold, in general.

Is \approx a congruence?

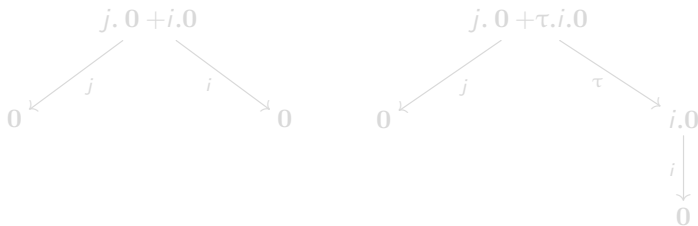
Example (initial τ restricts options 'menu')

$$i.0 \approx \tau.i.0$$

However

$$j.0 + i.0 \not\approx j.0 + \tau.i.0$$

Actually,



Is \approx a congruence?

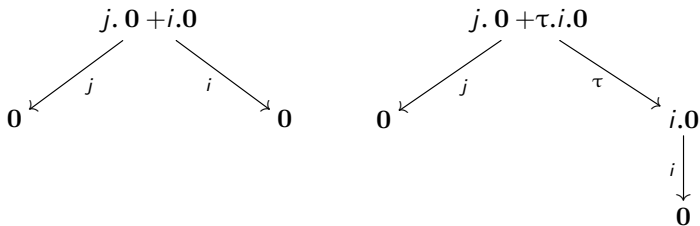
Example (initial τ restricts options 'menu')

$$i.0 \approx \tau.i.0$$

However

$$j.0 + i.0 \not\approx j.0 + \tau.i.0$$

Actually,



Forcing a congruence: $E = F$

Solution: force any **initial** τ to be matched by another τ

Process equality

Two processes E and F are **equal** (or **observationally congruent**) iff

- i) $E \approx F$
- ii) $E \xrightarrow{\tau} E' \Rightarrow F \xrightarrow{\tau} X \xRightarrow{\epsilon} F'$ and $E' \approx F'$
- iii) $F \xrightarrow{\tau} F' \Rightarrow E \xrightarrow{\tau} X \xRightarrow{\epsilon} E'$ and $E' \approx F'$

- note that $E \neq \tau.E$, but $\tau.E = \tau.\tau.E$

Forcing a congruence: $E = F$

Solution: force any **initial** τ to be matched by another τ

Process equality

Two processes E and F are **equal** (or **observationally congruent**) iff

i) $E \approx F$

ii) $E \xrightarrow{\tau} E' \Rightarrow F \xrightarrow{\tau} X \xRightarrow{\epsilon} F'$ and $E' \approx F'$

iii) $F \xrightarrow{\tau} F' \Rightarrow E \xrightarrow{\tau} X \xRightarrow{\epsilon} E'$ and $E' \approx F'$

- note that $E \neq \tau.E$, but $\tau.E = \tau.\tau.E$

Forcing a congruence: $E = F$

$=$ can be regarded as a restriction of \approx to all pairs of processes which preserve it in **additive** contexts

Lemma

Let E and F be processes st the union of their sorts is distinct of L . Then,

$$E = F \equiv \forall_{G \in \mathbb{P}}. (E + G \approx F + G)$$

Properties of $=$

Lemma

$$E \approx F \equiv (E = F) \vee (E = \tau.F) \vee (\tau.E = F)$$

- note that $E \neq \tau.E$, but $\tau.E = \tau.\tau.E$

Properties of $=$

Lemma

$$\sim \subseteq = \subseteq \approx$$

So,

the whole \sim theory remains valid

Additionally,

Lemma (additional laws)

$$a.\tau.E = a.E$$

$$E + \tau.E = \tau.E$$

$$a.(E + \tau.F) = a.(E + \tau.F) + a.F$$

Solving equations

Have equations over (\mathbb{P}, \sim) or $(\mathbb{P}, =)$ (unique) solutions?

Lemma

Recursive equations $\tilde{X} = \tilde{E}(\tilde{X})$ or $\tilde{X} \sim \tilde{E}(\tilde{X})$, over \mathbb{P} , have **unique** solutions (up to $=$ or \sim , respectively). Formally,

- i) Let $\tilde{E} = \{E_i \mid i \in I\}$ be a family of expressions with a maximum of I free variables ($\{X_i \mid i \in I\}$) such that any variable free in E_i is **weakly guarded**. Then

$$\tilde{P} \sim \{\tilde{P}/\tilde{X}\}\tilde{E} \wedge \tilde{Q} \sim \{\tilde{Q}/\tilde{X}\}\tilde{E} \Rightarrow \tilde{P} \sim \tilde{Q}$$

- ii) Let $\tilde{E} = \{E_i \mid i \in I\}$ be a family of expressions with a maximum of I free variables ($\{X_i \mid i \in I\}$) such that any variable free in E_i is **guarded** and **sequential**. Then

$$\tilde{P} = \{\tilde{P}/\tilde{X}\}\tilde{E} \wedge \tilde{Q} = \{\tilde{Q}/\tilde{X}\}\tilde{E} \Rightarrow \tilde{P} = \tilde{Q}$$

Solving equations

Have equations over (\mathbb{P}, \sim) or $(\mathbb{P}, =)$ **(unique) solutions?**

Lemma

Recursive equations $\tilde{X} = \tilde{E}(\tilde{X})$ or $\tilde{X} \sim \tilde{E}(\tilde{X})$, over \mathbb{P} , have **unique** solutions (up to $=$ or \sim , respectively). Formally,

- i) Let $\tilde{E} = \{E_i \mid i \in I\}$ be a family of expressions with a maximum of I free variables ($\{X_i \mid i \in I\}$) such that any variable free in E_i is **weakly guarded**. Then

$$\tilde{P} \sim \{\tilde{P}/\tilde{X}\}\tilde{E} \wedge \tilde{Q} \sim \{\tilde{Q}/\tilde{X}\}\tilde{E} \Rightarrow \tilde{P} \sim \tilde{Q}$$

- ii) Let $\tilde{E} = \{E_i \mid i \in I\}$ be a family of expressions with a maximum of I free variables ($\{X_i \mid i \in I\}$) such that any variable free in E_i is **guarded** and **sequential**. Then

$$\tilde{P} = \{\tilde{P}/\tilde{X}\}\tilde{E} \wedge \tilde{Q} = \{\tilde{Q}/\tilde{X}\}\tilde{E} \Rightarrow \tilde{P} = \tilde{Q}$$

Solving equations

Have equations over (\mathbb{P}, \sim) or $(\mathbb{P}, =)$ **(unique) solutions?**

Lemma

Recursive equations $\tilde{X} = \tilde{E}(\tilde{X})$ or $\tilde{X} \sim \tilde{E}(\tilde{X})$, over \mathbb{P} , have **unique** solutions (up to $=$ or \sim , respectively). Formally,

- i) Let $\tilde{E} = \{E_i \mid i \in I\}$ be a family of expressions with a maximum of I free variables $(\{X_i \mid i \in I\})$ such that any variable free in E_i is **weakly guarded**. Then

$$\tilde{P} \sim \{\tilde{P}/\tilde{X}\}\tilde{E} \wedge \tilde{Q} \sim \{\tilde{Q}/\tilde{X}\}\tilde{E} \Rightarrow \tilde{P} \sim \tilde{Q}$$

- ii) Let $\tilde{E} = \{E_i \mid i \in I\}$ be a family of expressions with a maximum of I free variables $(\{X_i \mid i \in I\})$ such that any variable free in E_i is **guarded** and **sequential**. Then

$$\tilde{P} = \{\tilde{P}/\tilde{X}\}\tilde{E} \wedge \tilde{Q} = \{\tilde{Q}/\tilde{X}\}\tilde{E} \Rightarrow \tilde{P} = \tilde{Q}$$

Conditions on variables

guarded :

X occurs in a sub-expression of type $a.E'$ for $a \in Act - \{\tau\}$

weakly guarded :

X occurs in a sub-expression of type $a.E'$ for $a \in Act$

in both cases assures that, until a guard is reached, behaviour does not depend on the process that instantiates the variable

example: X is **weakly guarded** in both $\tau.X$ and $\tau.0 + a.X + b.a.X$ but **guarded** only in the second

Conditions on variables

guarded :

X occurs in a sub-expression of type $a.E'$ for $a \in Act - \{\tau\}$

weakly guarded :

X occurs in a sub-expression of type $a.E'$ for $a \in Act$

in both cases assures that, until a guard is reached, behaviour does not depend on the process that instantiates the variable

example: X is **weakly guarded** in both $\tau.X$ and $\tau.0 + a.X + b.a.X$ but **guarded** only in the second

Conditions on variables

sequential :

X is sequential in E if every **strict** sub-expression in which X occurs is either $a.E'$, for $a \in \text{Act}$, or $\Sigma\tilde{E}$.

avoids X to become guarded by a τ as a result of an interaction

example: X is not **sequential** in $X = (\bar{a}.X \mid a.0) \setminus_{\{a\}}$

Conditions on variables

sequential :

X is sequential in E if every **strict** sub-expression in which X occurs is either $a.E'$, for $a \in \text{Act}$, or $\Sigma\tilde{E}$.

avoids X to become guarded by a τ as a result of an interaction

example: X is not **sequential** in $X = (\bar{a}.X \mid a.0) \setminus \{a\}$

Example (1)

Consider

$$Sem \hat{=} get.put.Sem$$

$$P_1 \hat{=} \overline{get}.c_1.\overline{put}.P_1$$

$$P_2 \hat{=} \overline{get}.c_2.\overline{put}.P_2$$

$$S \hat{=} (Sem \mid P_1 \mid P_2) \setminus_{\{get, put\}}$$

and

$$S' \hat{=} \tau.c_1.S' + \tau.c_2.S'$$

to prove $S \sim S'$, show both are solutions of

$$X = \tau.c_1.X + \tau.c_2.X$$

Example (1)

Consider

$$Sem \hat{=} get.put.Sem$$

$$P_1 \hat{=} \overline{get}.c_1.\overline{put}.P_1$$

$$P_2 \hat{=} \overline{get}.c_2.\overline{put}.P_2$$

$$S \hat{=} (Sem \mid P_1 \mid P_2) \setminus_{\{get, put\}}$$

and

$$S' \hat{=} \tau.c_1.S' + \tau.c_2.S'$$

to prove $S \sim S'$, show both are **solutions** of

$$X = \tau.c_1.X + \tau.c_2.X$$

Example (1)

proof

$$\begin{aligned}
 S &= \tau. (c_1.\overline{put}.P_1 \mid P_2 \mid put.Sem) \setminus_K + \tau.(P_1 \mid c_2.\overline{put}.P_2 \mid put.Sem) \setminus_K \\
 &= \tau.c_1. (\overline{put}.P_1 \mid P_2 \mid put.Sem) \setminus_K + \tau.c_2.(P_1 \mid \overline{put}.P_2 \mid put.Sem) \setminus_K \\
 &= \tau.c_1.\tau. (P_1 \mid P_2 \mid Sem) \setminus_K + \tau.c_2.\tau.(P_1 \mid P_2 \mid Sem) \setminus_K \\
 &= \tau.c_1.\tau.S + \tau.c_2.\tau.S \\
 &= \tau.c_1.S + \tau.c_2.S \\
 &= \{S/X\}E
 \end{aligned}$$

for S' is immediate

Example (2)

Consider,

$$B \hat{=} in.B_1$$

$$B_1 \hat{=} in.B_2 + \overline{out}.B$$

$$B_2 \hat{=} \overline{out}.B_1$$

$$B' \hat{=} (C_1 \mid C_2) \setminus m$$

$$C_1 \hat{=} in.\overline{m}.C_1$$

$$C_2 \hat{=} m.\overline{out}.C_2$$

B' is a solution of

$$X = E(X, Y, Z) = in.Y$$

$$Y = E_1(X, Y, Z) = in.Z + \overline{out}.X$$

$$Z = E_3(X, Y, Z) = \overline{out}.Y$$

through $\sigma = \{B/X, B_1/Y, B_2/Z\}$

Example (2)

To prove $B = B'$

$$\begin{aligned} B' &= (C_1 \mid C_2) \setminus_m \\ &= in.(\bar{m}.C_1 \mid C_2) \setminus_m \\ &= in.\tau.(C_1 \mid \overline{out}.C_2) \setminus_m \\ &= in.(C_1 \mid \overline{out}.C_2) \setminus_m \end{aligned}$$

Let $S_1 = (C_1 \mid \overline{out}.C_2) \setminus_m$ to proceed:

$$\begin{aligned} S_1 &= (C_1 \mid \overline{out}.C_2) \setminus_m \\ &= in.(\bar{m}.C_1 \mid \overline{out}.C_2) \setminus_m + \overline{out}.(C_1 \mid C_2) \setminus_m \\ &= in.(\bar{m}.C_1 \mid \overline{out}.C_2) \setminus_m + \overline{out}.B' \end{aligned}$$

Example (2)

Finally, let, $S_2 = (\overline{m}.C_1 \mid \overline{out}.C_2) \setminus_m$. Then,

$$\begin{aligned} S_2 &= (\overline{m}.C_1 \mid \overline{out}.C_2) \setminus_m \\ &= \overline{out}.(\overline{m}.C_1 \mid C_2) \setminus_m \\ &= \overline{out}.\tau.(C_1 \mid \overline{out}.C_2) \setminus_m \\ &= \overline{out}.\tau.S_1 \\ &= \overline{out}.S_1 \end{aligned}$$

Example (2)

Note the same problem can be solved with a system of 2 equations:

$$X = E(X, Y) = in.Y$$

$$Y = E'(X, Y) = in.\overline{out}.Y + \overline{out}.in.Y$$

Clearly, by substitution,

$$B = in.B_1$$

$$B_1 = in.\overline{out}.B_1 + \overline{out}.in.B_1$$

Example (2)

On the other hand, it's already proved that $B' = \dots = in.S_1$.
so,

$$\begin{aligned}
 S_1 &= (C_1 \mid \overline{out}.C_2) \setminus_m \\
 &= in.(\overline{m}.C_1 \mid \overline{out}.C_2) \setminus_m + \overline{out}.B' \\
 &= in.\overline{out}.(\overline{m}.C_1 \mid C_2) \setminus_m + \overline{out}.B' \\
 &= in.\overline{out}.\tau.(C_1 \mid \overline{out}.C_2) \setminus_m + \overline{out}.B' \\
 &= in.\overline{out}.\tau.S_1 + \overline{out}.B' \\
 &= in.\overline{out}.S_1 + \overline{out}.B' \\
 &= in.\overline{out}.S_1 + \overline{out}.in.S_1
 \end{aligned}$$

Hence, $B' = \{B'/X, S_1/Y\}E$ and $S_1 = \{B'/X, S_1/Y\}E'$