The π-calculus, syntax and semantics

Let us make it a bit more formal

- we have written *reductions* ($\longrightarrow$), leading from one process to another one

Let us make it a bit more formal

- we have written *reductions* ($\longrightarrow$), leading from one process to another one

- what is a process (a state of the system)?

Let us make it a bit more formal

- we have written *reductions* ($\longrightarrow$), leading from one process to another one

- what is a process (a state of the system)?
  a term? not exactly:

▷ $P \mid (Q \mid R)$ "=" $(Q \mid P) \mid R$
put subprocesses next to eachother so that they can interact

Let us make it a bit more formal

- we have written *reductions* ($\longrightarrow$), leading from one process to another one

- what is a process (a state of the system)?
  a term?   not exactly:

▷ $P \mid (Q \mid R)$ “$=$” $(Q \mid P) \mid R$
put subprocesses next to eachother so that they can interact

▷ $P \mid \mathbf{0}$ “$=$” P
erase $\mathbf{0}$ *produced along transitions*

Let us make it a bit more formal

- we have written *reductions* ($\longrightarrow$), leading from one process to another one

- what is a process (a state of the system)?
  a term?   not exactly:

▷ $P \mid (Q \mid R)$ "$=$" $(Q \mid P) \mid R$
put subprocesses next to eachother so that they can interact

▷ $P \mid \mathbf{0}$ "$=$" P
erase $\mathbf{0}$ *produced along transitions*

... "$=$" $\rightarrow$ $\boxed{\equiv}$, structural congruence

Structural congruence

$\equiv$ is the least congruence satisfying the following axioms:

parallel composition: commutative monoid

$$P \,|\, \mathbf{0} \equiv P \qquad P \,|\, Q \equiv Q \,|\, P \qquad P \,|\, (Q \,|\, R) \equiv (P \,|\, Q) \,|\, R$$

Structural congruence

$\equiv$ is the least congruence satisfying the following axioms:

parallel composition: commutative monoid

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

restriction

$$(\boldsymbol{\nu} n)\, \mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)\, P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)\, P$$

Structural congruence

$\equiv$ is the least congruence satisfying the following axioms:

parallel composition: commutative monoid

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

restriction

$$(\boldsymbol{\nu} n)\,\mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)\,P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)\,P$$

$$(\boldsymbol{\nu} n)\,(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)\,Q \quad \text{if } n \notin \mathsf{fn}(P)$$

Structural congruence

$\equiv$ is the least congruence satisfying the following axioms:

parallel composition: commutative monoid

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

restriction

$$(\boldsymbol{\nu} n)\, \mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)\, P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)\, P$$

$$(\boldsymbol{\nu} n)\,(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)\, Q \quad \text{if } n \notin \mathsf{fn}(P)$$

(restrictions "float around")

Structural congruence

$\equiv$ is the least congruence satisfying the following axioms:

parallel composition: commutative monoid

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

restriction

$$(\boldsymbol{\nu} n)\, \mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)\, P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)\, P$$

$$(\boldsymbol{\nu} n)\,(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)\, Q \quad \text{if } n \notin \mathsf{fn}(P)$$

(restrictions "float around")

replication

$$!P \equiv !P \mid P$$

Structural congruence

$\equiv$ is the least congruence satisfying the following axioms:

parallel composition: commutative monoid

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

restriction

$$(\boldsymbol{\nu} n)\, \mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)\, P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)\, P$$

$$(\boldsymbol{\nu} n)\,(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)\, Q \quad \text{if } n \notin \mathsf{fn}(P)$$

(restrictions "float around")

replication

$$!P \equiv !P \mid P$$

Remarks: $(\boldsymbol{\nu} x)\, P \equiv P$ if $x \notin \mathsf{fn}(P)$

$$P \equiv (\boldsymbol{\nu} \tilde{x})\,(M_1 \mid \ldots \mid M_k \mid !R_1 \mid \ldots \mid !R_n)$$

Evolution of states: reduction

let $\longrightarrow$ be the smallest relation satisfying the following axioms:

$$\overline{n}\langle a\rangle.P \mid n(b).Q \;\longrightarrow\; P \mid Q_{\{b\leftarrow a\}}$$

Evolution of states: reduction

let $\longrightarrow$ be the smallest relation satisfying the following axioms:

$$\overline{n}\langle a\rangle.P \mid n(b).Q \;\longrightarrow\; P \mid Q_{\{b\leftarrow a\}}$$

$$\frac{P \;\longrightarrow\; P' \quad P \equiv Q \quad P' \equiv Q'}{Q \;\longrightarrow\; Q'}$$

Evolution of states: reduction

let $\longrightarrow$ be the smallest relation satisfying the following axioms:

$$\overline{n}\langle a\rangle.P \mid n(b).Q \;\longrightarrow\; P \mid Q_{\{b\leftarrow a\}}$$

$$\frac{P \;\longrightarrow\; P' \quad P \equiv Q \quad P' \equiv Q'}{Q \;\longrightarrow\; Q'}$$

$$\frac{P \;\longrightarrow\; P'}{(\boldsymbol{\nu} n)\, P \;\longrightarrow\; (\boldsymbol{\nu} n)\, P'} \qquad \frac{P \;\longrightarrow\; P'}{P \mid Q \;\longrightarrow\; P' \mid Q}$$

$\rightarrow$ *reduction based* presentation of the operational semantics

Derivations: an example

Notation: $$\overline{x}(y) \quad = \quad (\boldsymbol{\nu} y)\, \overline{x}\langle y \rangle.\mathbf{0}$$

Derivations: an example

Notation: $\quad \overline{x}(y) \quad = \quad (\boldsymbol{\nu} y)\, \overline{x}\langle y\rangle.\mathbf{0}$

$$
\begin{array}{lll}
 & x(z).\overline{w}\langle z\rangle \mid !\overline{x}(y) & \\
\equiv & x(z).\overline{w}\langle z\rangle \mid \overline{x}(y) \mid !\overline{x}(y) & \text{unfold !} \\
\equiv & (\boldsymbol{\nu} y)\,(x(z).\overline{w}z \mid \overline{x}\langle y\rangle) \mid !\overline{x}(y) & \text{scope } \boldsymbol{\nu} \\
\longrightarrow & (\boldsymbol{\nu} y)\,(\overline{w}\langle y\rangle \mid \mathbf{0}) \mid !\overline{x}(y) & \text{communication + congr.} \\
\equiv & \overline{w}(y) \mid !\overline{x}(y) & \text{GC}
\end{array}
$$

Derivations: an example

Notation: $\quad \overline{x}(y) \quad = \quad (\boldsymbol{\nu} y)\, \overline{x}\langle y\rangle.\mathbf{0}$

$$
\begin{array}{lll}
 & x(z).\overline{w}\langle z\rangle \mid !\overline{x}(y) & \\
\equiv & x(z).\overline{w}\langle z\rangle \mid \overline{x}(y) \mid !\overline{x}(y) & \text{unfold !} \\
\equiv & (\boldsymbol{\nu} y)\,(x(z).\overline{w}z \mid \overline{x}\langle y\rangle) \mid !\overline{x}(y) & \text{scope } \boldsymbol{\nu} \\
\longrightarrow & (\boldsymbol{\nu} y)\,(\overline{w}\langle y\rangle \mid \mathbf{0}) \mid !\overline{x}(y) & \text{communication + congr.} \\
\equiv & \overline{w}(y) \mid !\overline{x}(y) & \text{GC}
\end{array}
$$

hence $\quad x(z).\overline{w}\langle z\rangle \mid !\overline{x}(y) \quad \longrightarrow \quad \overline{w}(y) \mid !\overline{x}(y)$

G. Berry, G. Boudol: "*The chemical abstract machine*", POPL'90

What is $\equiv$ intented to say?

- why should'nt we include $(\boldsymbol{\nu} c)\,\overline{c}\langle a\rangle.P \equiv$

What is $\equiv$ intented to say?

- why should'nt we include $(\boldsymbol{\nu} c)\,\overline{c}\langle a\rangle.P \equiv \mathbf{0}$?

What is $\equiv$ intented to say?

- why should'nt we include $(\boldsymbol{\nu} c)\, \overline{c}\langle a\rangle.P \equiv \mathbf{0}$?

$\equiv$ laws talk about *state*, as opposed to *behavioural laws*

What is $\equiv$ intented to say?

- why should'nt we include $(\boldsymbol{\nu} c)\, \overline{c}\langle a\rangle.P \equiv \mathbf{0}$?

$\equiv$ laws talk about *state*, as opposed to *behavioural laws*

- what about other laws then?

connections with the issue of *decidability* of $\equiv$

What is $\equiv$ intented to say?

- why should'nt we include $(\boldsymbol{\nu} c)\,\overline{c}\langle a\rangle.P \equiv \mathbf{0}$?

$\equiv$ laws talk about *state*, as opposed to *behavioural laws*

- what about other laws then?

connections with the issue of *decidability* of $\equiv$

$$!(P|Q) \equiv !P \mid !Q \qquad !!P \equiv !P \qquad !\mathbf{0} \equiv \mathbf{0}$$

What is $\equiv$ intented to say?

- why should'nt we include $(\boldsymbol{\nu} c)\, \overline{c}\langle a\rangle.P \equiv \mathbf{0}$?

$\equiv$ laws talk about *state*, as opposed to *behavioural laws*

- what about other laws then?

connections with the issue of *decidability* of $\equiv$

$$!(P|Q) \equiv !P \mid !Q \qquad !!P \equiv !P \qquad !\mathbf{0} \equiv \mathbf{0}$$
$$(\boldsymbol{\nu} x)\, M.P \equiv M.(\boldsymbol{\nu} x)\, P \quad \text{if } x \notin \mathsf{fn}(M)$$

cf. works by Engelfriet & Gelsema

Choice operator

the operator of *choice* is inherited from CCS

$P + Q$ behaves like P *or* Q

Choice operator

the operator of *choice* is inherited from CCS

$P + Q$ behaves like P *or* Q

$$\begin{array}{ccc} & \overline{\mathsf{pile}}\langle v\rangle + \overline{\mathsf{face}}\langle v\rangle \mid \mathsf{pile}(x).P \mid \mathsf{face}(x).Q & \\ \swarrow & & \searrow \\ P \mid \mathsf{face}(x).Q & & \mathsf{pile}(x).P \mid Q \end{array}$$

N.B.: $+$ has greater priority than $|$

Exercise: adding choice

What modifications should be made to the reduction-based presentation of the operational semantics of the π-calculus to include choice in the language?

"Example": alternating bit protocol

$$\begin{array}{rcl}
\mathit{Send} & \stackrel{\text{def}}{=} & !\mathsf{send}(b).\mathsf{acc}.\overline{\mathsf{trans}}\langle\neg b\rangle.\overline{\mathsf{wait}}\langle\neg b\rangle \\
\mathit{Wait} & \stackrel{\text{def}}{=} & !\mathsf{wait}(b).(\mathsf{ackn}(b').\overline{\mathsf{send}}\langle b'\rangle \\
 & & \qquad\qquad + \mathsf{loss}.\overline{\mathsf{trans}}\langle b\rangle.\overline{\mathsf{wait}}\langle b\rangle) \\
\mathit{Receive} & \stackrel{\text{def}}{=} & !\mathsf{rec}(b).\mathsf{trans}(b').\mathit{if}\ b_1 = b'_1 \\
 & & \quad \mathit{then}\ \overline{\mathsf{ackn}}\langle b\rangle \mid \overline{\mathsf{rec}}\langle b\rangle \\
 & & \quad \mathit{else}\ \overline{\mathsf{deli}}.(\overline{\mathsf{ackn}}\langle\neg b\rangle \mid \overline{\mathsf{rec}}\langle\neg b\rangle) \\
\mathit{Noise} & \stackrel{\text{def}}{=} & !\mathsf{noise}.(\,\mathsf{trans}(b).\overline{\mathsf{loss}}.\overline{\mathsf{noise}} + \mathsf{ackn}(b).\overline{\mathsf{loss}}.\overline{\mathsf{noise}}\,) \\
\mathit{System} & \stackrel{\text{def}}{=} & (\boldsymbol{\nu}\,\mathsf{trans}, \mathsf{ackn}, \mathsf{send}, \mathsf{wait}, \mathsf{rec}, \mathsf{loss}, \mathsf{noise}, b) \\
 & & \quad (\,\mathit{Send} \mid \mathit{Wait} \mid \mathit{Receive} \mid \mathit{Noise} \mid !b\,\mathsf{true} \mid \\
 & & \qquad \mid \overline{\mathsf{send}}\langle b\rangle \mid \overline{\mathsf{rec}}\langle b\rangle \mid \overline{\mathsf{noise}}\,) \\
\mathit{Specif} & \stackrel{\text{def}}{=} & (\boldsymbol{\nu}c)\,(!c.\mathsf{acc}.\overline{\mathsf{deli}}.\overline{c} \mid \overline{c})
\end{array}$$

Choice and definitions

- the choice operator is useful for specification purposes

- *recursive definitions* can be used instead of replication: write

$$A[\tilde{x}] \quad = \quad P\,,$$

where P may contain subterms of the form $A[\tilde{y}]$

Recursive definitions: exercises

- a specification (using mutually recursive definitions):

$$
\begin{aligned}
B_0[x, y, z] &\stackrel{\text{def}}{=} y(w).B_1[x, y, z, w] \; + \; x(u).B_0[x, u, z] \\
B_1[x, y, z, w] &\stackrel{\text{def}}{=} y(v).B_2[x, y, z, w, v] \; + \; \overline{z}\langle w \rangle.B_0[x, y, z] \\
B_2[x, y, z, w, v] &\stackrel{\text{def}}{=} \overline{z}\langle w \rangle.B_1[x, y, z, v]
\end{aligned}
$$

what's that?

Recursive definitions: exercises

- a specification (using mutually recursive definitions):

$$
\begin{aligned}
B_0[x, y, z] &\stackrel{\text{def}}{=} y(w).B_1[x, y, z, w] \;+\; x(u).B_0[x, u, z] \\
B_1[x, y, z, w] &\stackrel{\text{def}}{=} y(v).B_2[x, y, z, w, v] \;+\; \overline{z}\langle w\rangle.B_0[x, y, z] \\
B_2[x, y, z, w, v] &\stackrel{\text{def}}{=} \overline{z}\langle w\rangle.B_1[x, y, z, v]
\end{aligned}
$$

what's that?

- how can we encode recursive definitions in a π-calculus with replication?