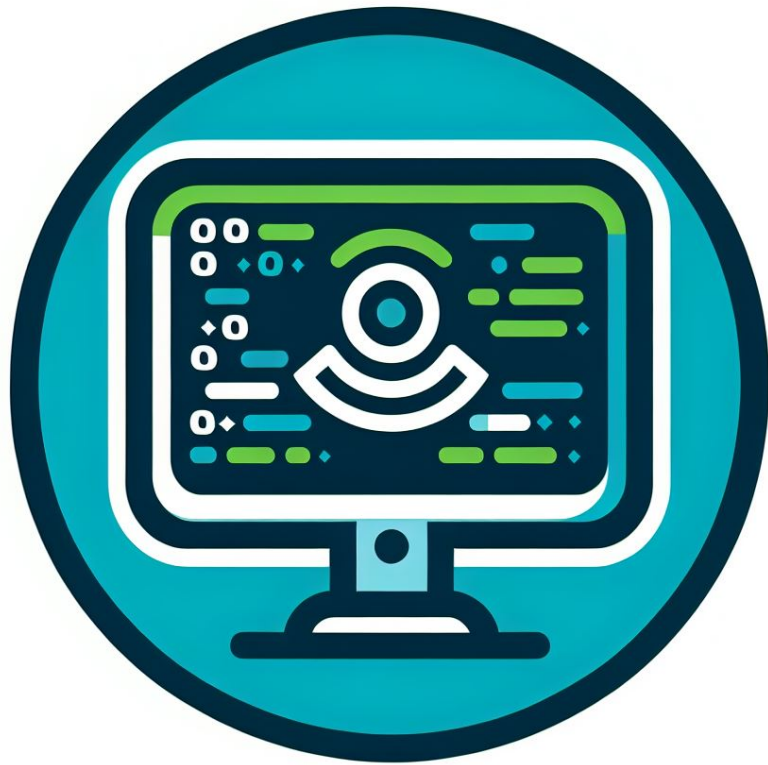




PSC 2024/25 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni

<http://www.di.unipi.it/~bruni/>

[http://didawiki.di.unipi.it/doku.php/
magistraleinformatica/psc/start](http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/start)

24 - Pi-calculus

π -calculus

Name mobility

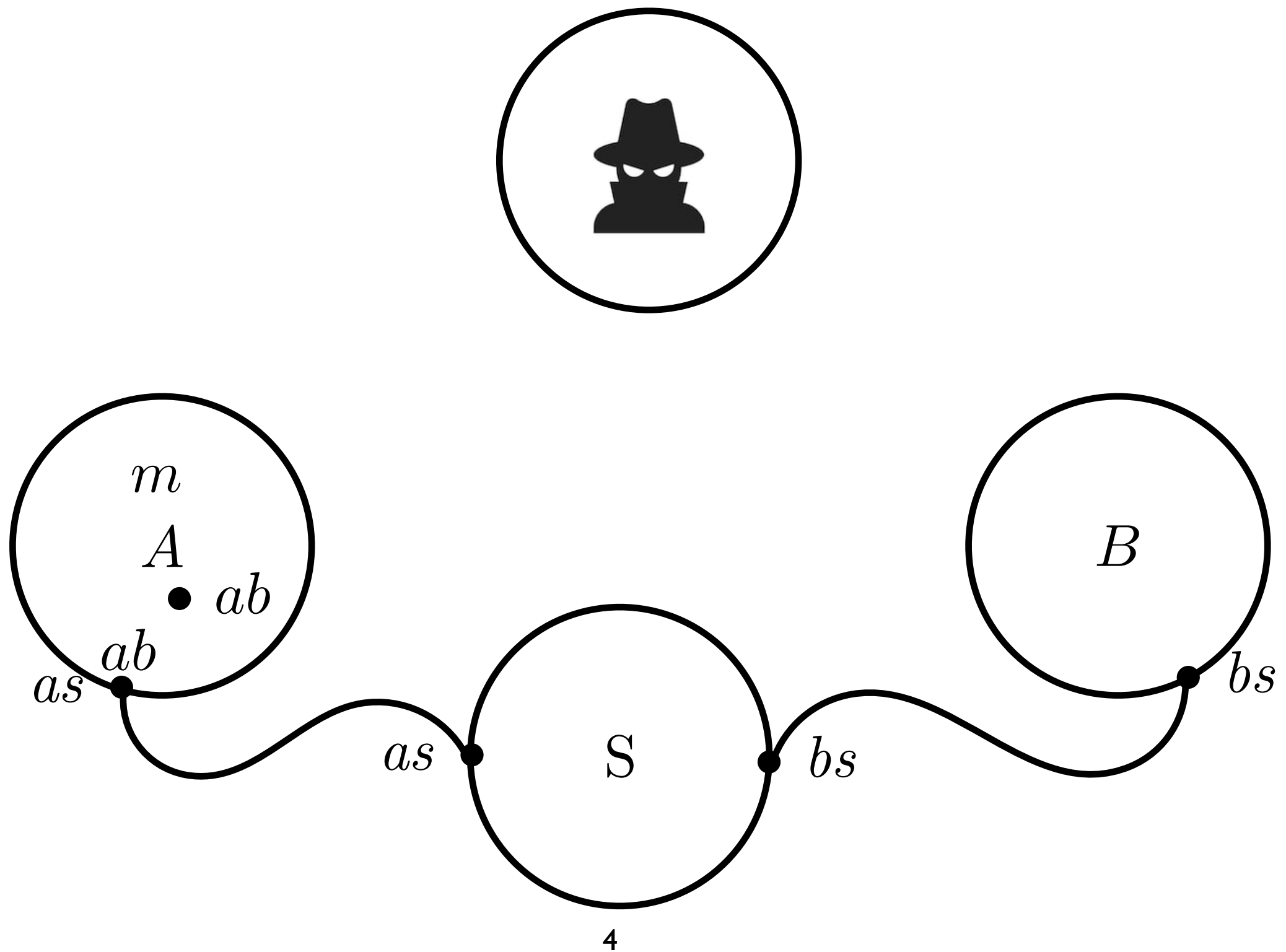
in CCS communication is statically defined

(channels cannot be transmitted / received)

what if channels are treated as ordinary values?

can we extend CCS to communicate communication means?

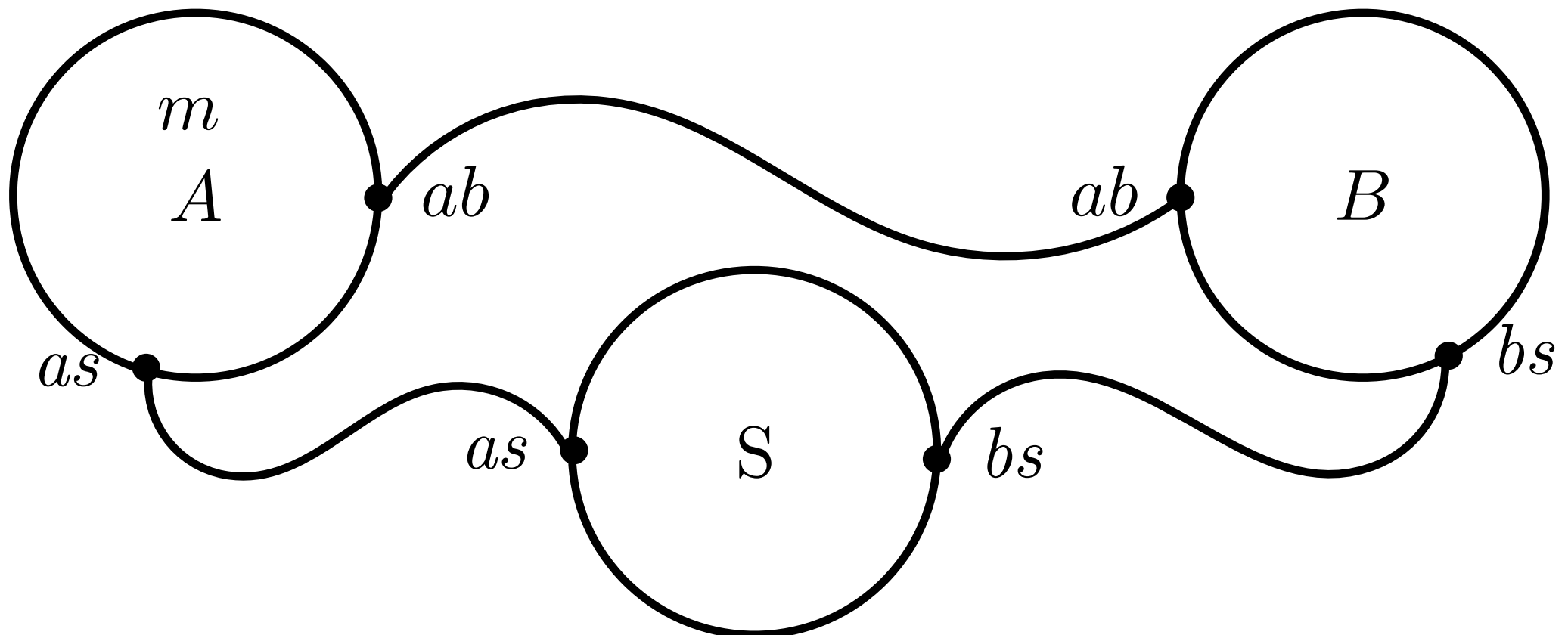
Name mobility



Name mobility

$(\boxed{A \backslash ab} \mid B \mid S) \backslash as \backslash bs$

$(\boxed{(A \mid B) \backslash ab} \mid S) \backslash as \backslash bs$



pi-calculus: syntax

π	$::=$	$\bar{x}y$	send y on x
		$x(y)$	receive on x and store in y
		τ	internal action

p	$::=$	nil	inactive process
		$\pi.p$	action prefix
		$[x = y]p$	matching (like, if $x=y$ then p)
		$p + p$	nondeterministic choice
		$p p$	parallel composition
		$(y)p$	restriction (like $p \backslash y$, also written $(\nu y)p$)
		$!p$	replication (like $p p p \dots$)

pi-calculus: the idea

like
 $y \leftarrow x$

like
 $x \leftarrow z$

$$x(y) . p \mid \bar{x}z . q \xrightarrow{\tau} p[z/y] \mid q$$

communication

like
 $z = \text{make}(\text{chan} \dots)$

$$x(y) . p \mid (z) \bar{x}z . q \xrightarrow{\tau} (z) p[z/y] \mid q$$

extrusion

pi-calculus: syntax

the only binders are input prefix and name restriction

$$x(y).p$$

$$(y)p$$

the notion of free and bound names are defined as usual

$$\text{fn}(p)$$

$$\text{bn}(p)$$

the notion of capture avoiding substitution is defined as usual

$$p[y/x]$$

processes are taken up-to alpha-renaming of bound names

$$x(y).p = x(z).(p[z/y]) \text{ if } z \notin \text{fn}(p)$$

$$(y)p = (z)(p[z/y]) \text{ if } z \notin \text{fn}(p)$$

pi-calculus: notation

write $\sum_{i=1}^n p_i$ instead of $p_1 + \cdots + p_n$

write $\prod_{i=1}^n p_i$ instead of $p_1 | \cdots | p_n$

write $(\tilde{a})p$ instead of $(a_1) \cdots (a_n)p$

input guarded replication: $!x(y).p$

pi-calculus: conventions

we omit trailing **nil**

$$(y)\bar{x}y.\bar{y}m.\mathbf{nil} \quad \text{as} \quad (y)\bar{x}y.\bar{y}m$$

vacuous messages written as CCS prefixes

$$\bar{x} \mid x.p + x.q$$

mismatch not necessary (if the number of cases is finite)

$$[x \neq y_i].p \quad \text{as} \quad \sum_{j \neq i} [x = y_j]p$$

polyadic communication (can be encoded)

$$\bar{x}\langle z_1, \dots, z_n \rangle.p \qquad x(y_1, \dots, y_n).q$$

$$\bar{x}\langle \widetilde{z} \rangle.p \qquad x(\widetilde{y}).q$$

Examples

$A \triangleq (ab) \bar{as} ab . \bar{ab} m . A'$

————— do something else
 ————— send m on ab
 ————— send ab on as
 ————— create a new (private) channel ab

```
ab := make(chan string)
as <- ab // send ab to the server
ab <- m  // send m on ab
```

$S \triangleq ! as(x) . \bar{bs} x$

————— omit **nil**
 ————— receive on as and forward to on bs

```
for {
  x := <-as
  bs <- x
}
```

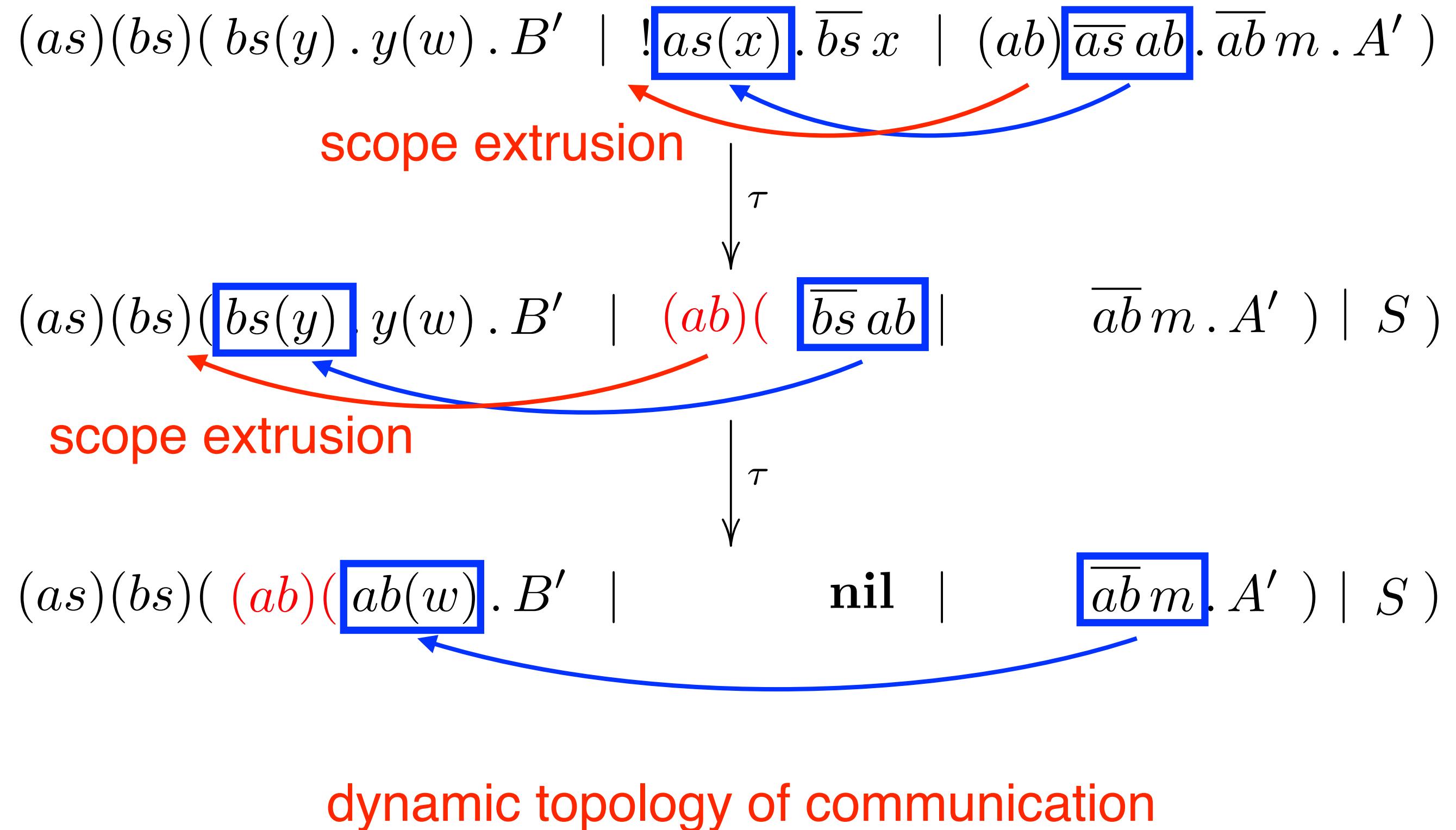
$B \triangleq bs(y) . y(w) . B'$

————— do something else
 ————— receive on y
 ————— receive y on bs

```
y := <-bs
w := <-y
```

$(as) (bs) (A \mid S \mid B)$

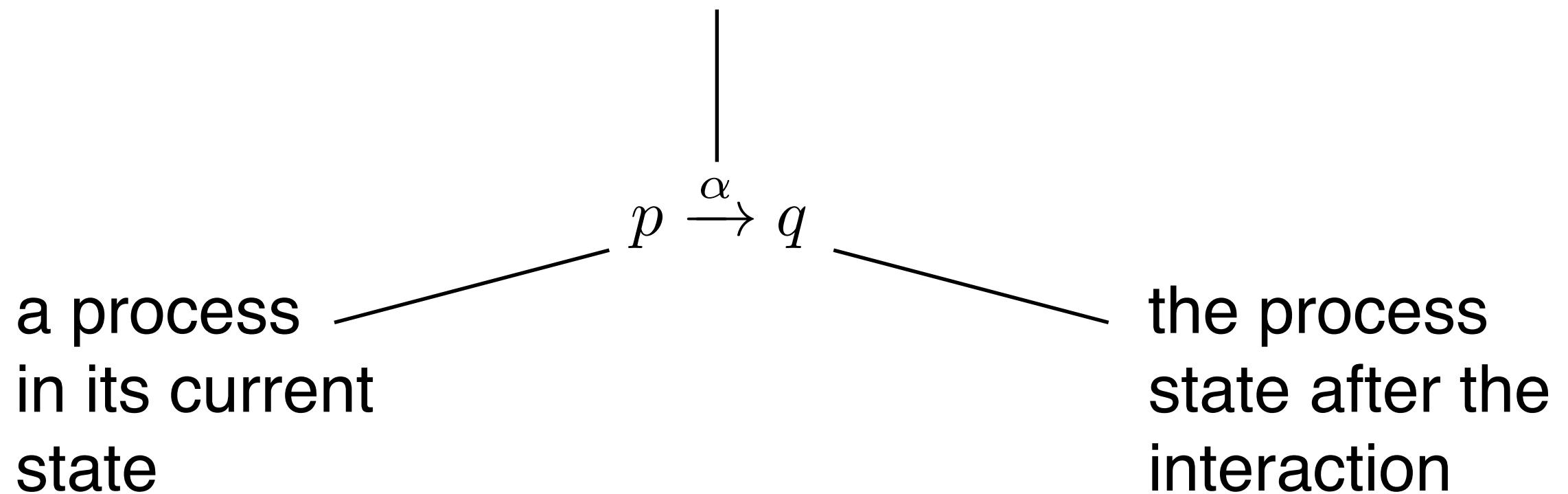
Example



π -calculus operational semantics

pi-calculus: LTS

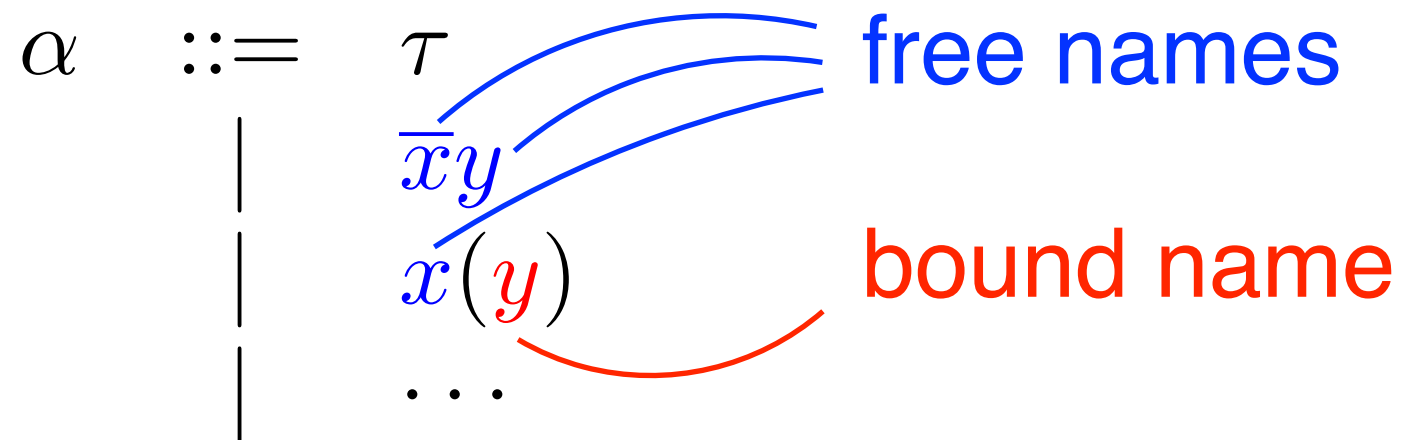
ongoing interaction
with the environment
(with other processes)



small-step semantics

pi-calculus labels

What is a label α ?



Action prefix

$$\text{Act}) \frac{}{\pi.p \xrightarrow{\pi} p}$$

note: $x(y).p \xrightarrow{x(y)} p$

$$x(y).p = x(z).(p[z/y]) \text{ if } z \notin \text{fn}(p)$$

$$x(y).p \xrightarrow{x(z)} p[z/y] \text{ if } z \notin \text{fn}(p)$$

we call *fresh* such z

z is just a placeholder, not an actual value

we define some sort of symbolic semantics for input
(the continuation needs an actual name to evolve)

Matching & Choice

$$\text{Match) } \frac{p \xrightarrow{\alpha} q}{[x = x] p \xrightarrow{\alpha} q}$$

example: $\text{login}(y) \cdot [y = \text{pwd}] p$

$$\text{SumL) } \frac{p_1 \xrightarrow{\alpha} q}{p_1 + p_2 \xrightarrow{\alpha} q}$$

$$\text{SumR) } \frac{p_2 \xrightarrow{\alpha} q}{p_1 + p_2 \xrightarrow{\alpha} q}$$

example: $\text{confirm}(y) \cdot ([y = \text{ok}] p + [y = \text{no}] q)$

Parallel

$$\text{ParL)} \frac{p_1 \xrightarrow{\alpha} q_1}{p_1 | p_2 \xrightarrow{\alpha} q_1 | p_2} \text{bn}(\alpha) \cap \text{fn}(p_2) = \emptyset$$

$$\text{ParR)} \frac{p_2 \xrightarrow{\alpha} q_2}{p_1 | p_2 \xrightarrow{\alpha} p_1 | q_2} \text{bn}(\alpha) \cap \text{fn}(p_1) = \emptyset$$

$$x(y) . \bar{y}z \mid w(u) \xrightarrow{x(v)} \bar{v}z \mid w(u)$$

fresh

$$x(y) . \bar{y}z \mid w(u) \xrightarrow{x(w)} \bar{w}z \mid w(u)$$

not fresh! not taken in input!

Communication

$$\text{ComL)} \frac{p_1 \xrightarrow{\bar{x}z} q_1 \quad p_2 \xrightarrow{x(y)} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | (q_2[z/y])}$$

$$\text{ComR)} \frac{p_1 \xrightarrow{x(y)} q_1 \quad p_2 \xrightarrow{\bar{x}z} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1[z/y] | q_2}$$

$$\begin{array}{ccc} \bar{x}z . y(v) & | & x(y) . \bar{y}z \xrightarrow{\tau} y(v) | \bar{z}z \\ \downarrow \bar{x}z & & \downarrow x(y) \\ y(v) & & \bar{y}z \end{array}$$

Restriction

$$\text{Res)} \frac{p \xrightarrow{\alpha} q}{(y)p \xrightarrow{\alpha} (y)q} \quad y \notin \text{fn}(\alpha) \cup \text{bn}(\alpha)$$

$$\text{Open)} \frac{p \xrightarrow{\bar{x}y} q}{(y)p \xrightarrow{\bar{x}(z)} q[z/y]} \quad \begin{array}{l} y \neq x \\ z \notin \text{fn}((y)p) \end{array}$$

a new kind of label:
bound output

we temporarily remove the restriction

$$\alpha ::= \begin{array}{|l} \tau \\ \bar{x}y \\ x(y) \\ \bar{x}(y) \end{array}$$

$$(y)\bar{x}y . \bar{y}w \xrightarrow{\bar{x}(y)} \bar{y}w$$

Scope extrusion

$$\text{CloseL)} \frac{p_1 \xrightarrow{\bar{x}(z)} q_1 \quad p_2 \xrightarrow{x(z)} q_2}{p_1 | p_2 \xrightarrow{\tau} (z)(q_1 | q_2)} \quad \text{CloseR)} \frac{p_1 \xrightarrow{x(z)} q_1 \quad p_2 \xrightarrow{\bar{x}(z)} q_2}{p_1 | p_2 \xrightarrow{\tau} (z)(q_1 | q_2)}$$

we restore the restriction

$$x(v) . v(b) \mid (y) \bar{x}y . \bar{y}a \xrightarrow{\tau} (z)(z(b) \mid \bar{z}a)$$

$$\begin{array}{ccc} & \downarrow x(z) & \downarrow \bar{x}(z) \\ & z(b) & \bar{z}a \end{array}$$

Replication

$$\text{Rep)} \frac{p|!p \xrightarrow{\alpha} q}{!p \xrightarrow{\alpha} q}$$

alternatively (to avoid infinite branching):

$$\text{Rep)} \frac{p|p \xrightarrow{\alpha} q}{!p \xrightarrow{\alpha} q|!p}$$

alternatively (if only input guarded replication is allowed):

$$\text{Rep)} \frac{p \xrightarrow{\alpha} q}{!p \xrightarrow{\alpha} q|!p}$$

$$!req(y).p \xrightarrow{req(y)} p|!req(y).p$$

pi-calculus: semantics

$$\begin{array}{ll}
 \text{Act)} \frac{}{\pi.p \xrightarrow{\pi} p} & \text{Match)} \frac{p \xrightarrow{\alpha} q}{[x = x]p \xrightarrow{\alpha} q} \quad \text{SumL)} \frac{p_1 \xrightarrow{\alpha} q}{p_1 + p_2 \xrightarrow{\alpha} q} \quad \text{SumR)} \dots \\
 \\
 \text{Rep)} \frac{p|!p \xrightarrow{\alpha} q}{!p \xrightarrow{\alpha} q} & \text{ParL)} \frac{p_1 \xrightarrow{\alpha} q_1}{p_1 | p_2 \xrightarrow{\alpha} q_1 | p_2} \quad \text{bn}(\alpha) \cap \text{fn}(p_2) = \emptyset \quad \text{ParR)} \dots \\
 \\
 \text{Res)} \frac{p \xrightarrow{\alpha} q}{(y)p \xrightarrow{\alpha} (y)q} \quad y \notin \text{fn}(\alpha) \cup \text{bn}(\alpha) & \text{ComL)} \frac{p_1 \xrightarrow{\bar{x}z} q_1 \quad p_2 \xrightarrow{x(y)} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | (q_2[z/y])} \quad \text{ComR)} \dots \\
 \\
 \text{Open)} \frac{p \xrightarrow{\bar{x}y} q}{(y)p \xrightarrow{\bar{x}(z)} q[z/y]} \quad \begin{array}{l} y \neq x \\ z \notin \text{fn}((y)p) \end{array} & \text{CloseL)} \frac{p_1 \xrightarrow{\bar{x}(z)} q_1 \quad p_2 \xrightarrow{x(z)} q_2}{p_1 | p_2 \xrightarrow{\tau} (z)(q_1 | q_2)} \quad \text{CloseR)} \dots
 \end{array}$$

Process constants

$$\Delta = \left\{ \begin{array}{lcl} A_1(\tilde{x}_1) & \triangleq & p_1 \\ \dots & & \\ A_n(\tilde{x}_n) & \triangleq & p_n \end{array} \right\} \quad \begin{array}{l} p ::= \dots \\ | A\langle \tilde{y} \rangle \text{ invocation} \end{array}$$

$$\frac{A(\tilde{x}) \triangleq p \in \Delta \quad p[\tilde{y}/\tilde{x}] \xrightarrow{\alpha} q}{A\langle \tilde{y} \rangle \xrightarrow{\alpha} q}$$

example $A(x) \triangleq x(y).B\langle y \rangle \quad B(y) \triangleq (z)\bar{y}z.A\langle z \rangle$

$$A\langle x \rangle \mid B\langle x \rangle \xrightarrow{\tau} (z_1)(B\langle z_1 \rangle \mid A\langle z_1 \rangle)$$

$$\xrightarrow{\tau} (z_1)(z_2)(A\langle z_2 \rangle \mid B\langle z_2 \rangle)$$

$$\xrightarrow{\tau} (z_1)(z_2)(z_3)(B\langle z_3 \rangle \mid A\langle z_3 \rangle)$$

Constants vs replication

$$\Delta = \left\{ \begin{array}{ccc} A_1(\tilde{x}_1) & \triangleq & p_1 \\ \dots & & \\ A_n(\tilde{x}_n) & \triangleq & p_n \end{array} \right\} \quad \begin{array}{l} p ::= \dots \\ | A\langle \tilde{y} \rangle \text{ invocation} \end{array}$$

introduce one dedicated channel a_i for each constant A_i

$$D \triangleq \prod_{i=1}^n !a_i(\tilde{x}_i). \hat{p}_i \quad (\text{requires polyadic communication})$$

$$\text{where } \hat{p} \triangleq p[\bar{a}_1 / A_1, \dots, \bar{a}_n / A_n]$$

then run $(\tilde{a})(\hat{q} | D)$ instead of q

Example

$$A(x) \triangleq x(y).B\langle y \rangle \qquad B(y) \triangleq (z)\bar{y}z.A\langle z \rangle$$

$$D \triangleq \prod_{i=1}^n !a_i(\tilde{x}_i).\hat{p}_i \qquad \text{where } \hat{p} \triangleq p[\bar{a}_1 / A_1, \dots, \bar{a}_n / A_n]$$

$$D \triangleq !a(x).x(y).\bar{b}y \mid !b(y).(z)\bar{y}z.\bar{a}z$$

$$\text{then run } (\tilde{a})(\hat{q} \mid D) \text{ instead of } q \qquad q = A\langle x \rangle \mid B\langle x \rangle$$

$$\begin{aligned} (a, b)(\bar{a}x \mid \bar{b}x \mid D) &\xrightarrow{\tau} (a, b)(x(y).\bar{b}y \mid \bar{b}x \mid D) \\ &\xrightarrow{\tau} (a, b)(x(y).\bar{b}y \mid (z)\bar{x}z.\bar{a}z \mid D) \\ &\xrightarrow{\tau} (a, b, z_1)(\bar{b}z_1 \mid \bar{a}z_1 \mid D) \end{aligned}$$



Exercise

what's wrong with the following encoding of matching?

$$[x = y]p \quad \text{as} \quad \bar{x} \mid y.p$$

there can be interferences!

other processes can steal the message on x

other processes can send messages on y

π -calculus bisimulation

Strong early ground bis

$$\mathbf{R} \subseteq \mathcal{P} \times \mathcal{P}$$

$$p \mathbf{R} q \Rightarrow \left\{ \begin{array}{l} \begin{array}{l} p \xrightarrow{\tau} p' \quad \Rightarrow \exists q'. q \xrightarrow{\tau} q' \wedge p' \mathbf{R} q' \\ p \xrightarrow{\bar{x}y} p' \quad \Rightarrow \exists q'. q \xrightarrow{\bar{x}y} q' \wedge p' \mathbf{R} q' \\ p \xrightarrow[\substack{y \notin \text{fn}(q)}]{\bar{x}(y)} p' \quad \Rightarrow \exists q'. q \xrightarrow{\bar{x}(y)} q' \wedge p' \mathbf{R} q' \\ p \xrightarrow[\substack{y \notin \text{fn}(q)}]{x(y)} p' \quad \Rightarrow \forall z. \exists q'. q \xrightarrow{x(y)} q' \wedge \boxed{p'[z/y] \mathbf{R} q'[z/y]} \end{array} \\ \text{and vice versa} \end{array} \right.$$

for every possible input

Strong early ground bis

$$\mathbf{R} \subseteq \mathcal{P} \times \mathcal{P}$$

$$p \mathbf{R} q \Rightarrow \left\{ \begin{array}{l} \boxed{\begin{array}{l} p \xrightarrow{\alpha} p' \\ \text{bn}(\alpha) \cap \text{fn}(q) = \emptyset \Rightarrow \exists q'. q \xrightarrow{\alpha} q' \wedge p' \mathbf{R} q' \\ \alpha \neq x(y) \end{array}} \\ \\ p \xrightarrow[y \notin \text{fn}(q)]{x(y)} p' \Rightarrow \forall z. \exists q'. q \xrightarrow{x(y)} q' \wedge p'[z/y] \mathbf{R} q'[z/y] \\ \text{and vice versa} \end{array} \right.$$

for every possible input

Strong *early* ground bis

$$\mathbf{R} \subseteq \mathcal{P} \times \mathcal{P}$$

$$p \mathbf{R} q \Rightarrow \left\{ \begin{array}{l} p \xrightarrow{\alpha} p' \\ \text{bn}(\alpha) \cap \text{fn}(q) = \emptyset \Rightarrow \exists q'. q \xrightarrow{\alpha} q' \wedge p' \mathbf{R} q' \\ \alpha \neq x(y) \\ \\ p \xrightarrow[y \notin \text{fn}(q)]{x(y)} p' \Rightarrow \boxed{\forall z.} \exists q'. q \xrightarrow{x(y)} q' \wedge p'[z/y] \mathbf{R} q'[z/y] \end{array} \right.$$

and vice versa

early quantification

Strong **late** ground bis

$$\mathbf{R} \subseteq \mathcal{P} \times \mathcal{P}$$

$$p \mathbf{R} q \Rightarrow \left\{ \begin{array}{l} p \xrightarrow{\alpha} p' \\ \text{bn}(\alpha) \cap \text{fn}(q) = \emptyset \Rightarrow \exists q'. q \xrightarrow{\alpha} q' \wedge p' \mathbf{R} q' \\ \alpha \neq x(y) \\ \\ p \xrightarrow[y \notin \text{fn}(q)]{x(y)} p' \Rightarrow \exists q'. q \xrightarrow{x(y)} q' \wedge \boxed{\forall z.} p'[z/y] \mathbf{R} q'[z/y] \\ \\ \text{and vice versa} \end{array} \right.$$

late quantification

Early vs late

early for every input Bob can choose a different move

$$\forall w. \exists q'. \dots$$

late Bob must choose one move, valid for every input

$$\exists q'. \forall w. \dots$$

Early/late bisimilarities

early bisimilarity

$p \sim_E q$ iff $\exists$ a strong early ground bisimulation $\mathbf{R}$ with $p \mathbf{R} q$

late bisimilarity

$p \sim_L q$ iff $\exists$ a strong late ground bisimulation $\mathbf{R}$ with $p \mathbf{R} q$

Early vs late

$$P \triangleq x(y) . \tau + x(y)$$

$$Q \triangleq P + x(y) . [y = a]\tau$$

$$P \xrightarrow{x(z)} \tau \xrightarrow{\tau} \mathbf{nil}$$

$$Q \xrightarrow{x(z)} \tau \xrightarrow{\tau} \mathbf{nil}$$

$$P \xrightarrow{x(z)} \mathbf{nil}$$

$$Q \xrightarrow{x(z)} \mathbf{nil}$$

z fresh

$$Q \xrightarrow{x(z)} [z = a]\tau$$

z fresh

early game

Alice picks $Q \xrightarrow{x(z)} [z = a]\tau$

for $z = a$ Bob picks $P \xrightarrow{x(z)} \tau$

for $z \neq a$ Bob picks $P \xrightarrow{x(z)} \mathbf{nil}$

Bob wins

$$P \sim_E Q$$

Early vs late

$$P \triangleq x(y) . \tau + x(y)$$

$$P \xrightarrow{x(z)} \tau \xrightarrow{\tau} \mathbf{nil}$$

$$P \xrightarrow{x(z)} \mathbf{nil}$$

z fresh

$$Q \triangleq P + x(y) . [y = a]\tau$$

$$Q \xrightarrow{x(z)} \tau \xrightarrow{\tau} \mathbf{nil}$$

$$Q \xrightarrow{x(z)} \mathbf{nil}$$

$$Q \xrightarrow{x(z)} [z = a]\tau$$

z fresh

late game

Alice picks $Q \xrightarrow{x(z)} [z = a]\tau$

if Bob picks $P \xrightarrow{x(z)} \tau$

if Bob picks $P \xrightarrow{x(z)} \mathbf{nil}$

Alice wins when $z \neq a$

Alice wins when $z = a$

Alice wins

$$P \not\sim_L Q$$

Early vs late: properties

$$P \sim_L Q \Rightarrow P \sim_E Q$$

$$P \sim_E Q \not\Rightarrow P \sim_L Q$$

(see previous example)

$\sim_E$ (resp. $\sim_L$) is a strong early (resp. late) ground bisimulation

$\sim_E$ and $\sim_L$ are equivalences but not congruences

(not preserved by input prefix)

$p \simeq_E q$ iff $p\sigma \sim_E q\sigma$ for every substitution σ

$p \simeq_L q$ iff $p\sigma \sim_L q\sigma$ for every substitution σ

$\simeq_E$ and $\simeq_L$ are congruences

(the largest congruences included in each bisimilarity)

Early & late: properties

$$p + \mathbf{nil} \simeq p$$

$$p + q \simeq q + p$$

$$p + (q + r) \simeq (p + q) + r$$

$$p + p \simeq p$$

$$p|\mathbf{nil} \simeq p$$

$$p|q \simeq q|p$$

$$p|(q|r) \simeq (p|q)|r$$

$$(x)\mathbf{nil} \simeq \mathbf{nil}$$

$$(x)(y)p \simeq (y)(x)p$$

$$(x)(x)p \simeq (x)p$$

$$(x)(p + q) \simeq (x)p + (x)q$$

$$(x)(p|q) \simeq p|(x)q \text{ if } x \notin \text{fn}(p)$$

$$[x = x]p \simeq p$$

$$!p \simeq p|!p$$

Weak semantics

weak transitions: $p \xRightarrow{\alpha} q$ defined as expected

weak bisimulations: Bob replies with weak moves

weak bisimilarities: $\approx_E$ and $\approx_L$ defined as expected

$\approx_E$ and $\approx_L$ are weak bisimulations

$\approx_E$ and $\approx_L$ are equivalences but not congruences

(not preserved by input prefix and choice)

weak observational congruences:

$p \cong_E q$ iff $\forall \sigma. p\sigma \approx_E q\sigma$ and $\forall r. p + r \approx_E q + r$

$p \cong_L q$ iff $\forall \sigma. p\sigma \approx_L q\sigma$ and $\forall r. p + r \approx_L q + r$