AN INTRODUCTION TO THE π-CALCULUS

(FROM HANDBOOK OF PROCESS ALGEBRA, BY JOACHIM PARROW)

We start from CCS:

$$P ::= 0 \mid \alpha.P \mid P+P \mid P|P \mid P[f] \mid P\backslash L \mid K$$

Action names (N): $\alpha \in N$ $f: N \to N$ $L \subseteq N$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1+P_2 \xrightarrow{\alpha} P_1'} + \text{Sym} \qquad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1|P_2 \xrightarrow{\alpha} P_1'|P_2} + \text{Sym}$$

$$\frac{P_1 \xrightarrow{\alpha} P_1' \quad P_2 \xrightarrow{\overline{\alpha}} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'} \qquad \frac{P \xrightarrow{\alpha} P' \quad \alpha, \overline{\alpha} \notin L}{P\backslash L \xrightarrow{\alpha} P'\backslash L} \qquad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'} \; K \triangleq P$$

Example: Pizza scenario

Customer $C \triangleq \overline{askPizza}.\overline{pay}.pizza$

Pizzaiolo $P \triangleq askPizza.pay.\overline{pizza}.P$

With Value Passing:

$C \triangleq \overline{askPizza}\langle margherita\rangle.\overline{pay}\langle 5€\rangle.pizza$

$P \triangleq askPizza(x).pay(y).$ if $y = price(x)$ then $\overline{pizza}.P$ else if $y > price(x)$ then $\overline{pizza}.\overline{refund}\langle y-price(x)\rangle$ else $\overline{askMoney}$ ....

In π-calculus: home delivery pizza!
The customer communicate the address where he wants the pizza be delivered.

π-calculus: formalism for mobility scenarios where processes are linked and such interconnections change as processes interact.
Basic concept: transfer of a comm. link for interaction with other ~~parties~~ parties via a computational step.

Why π-calculus is more expressive than CCS+value passing?
It allows migration of local scopes:
CCS: $(P|Q)\backslash a$ no other processes than P and Q can use a
π-calc: $(\nu a)(P|Q)$ a can be communicated and its scope extended or migrated

EXAMPLE: PRINTER ~~FROM~~ PARROW (*) RETRO
SYNTAX π-calculus

We assume a set of names N: a, b, .., x ..
that can be channels (communication ports), values, variables and booleans, references, objects, ...

$$P ::= 0 \mid \alpha.P \mid P+P \mid P|P$$
$$\mid \text{if } x=y \text{ then } P \mid \text{if } x\neq y \text{ then } P$$ (Match/Mismatch)
$$\mid (\nu x)P \mid K$$

behave like P if x=y, otherwise does nothing (nil) (**) RETRO

name x is local to P

$$\alpha ::= \bar{a}x \mid \text{[illegible]} \; a(x) \mid \tau$$

no interaction with the environment

NEW NAMES: x cannot be used to communicate with the environment

x is a placeholder for the receiv. name (VARIABLE)

2

STRUCTURAL CONGRUENCE

To represent all possible interactions with few operational rules, we use a structural congruence.

~~E.g.~~, the order in which we compose in parallel/sum processes should not matter.

$$\frac{P =_\alpha Q}{P \equiv Q}$$

Q can be obtained from P by a finite number of changes of bound names

$x(z).P$, $(\nu z)P$ — BOUND ~~FREE~~ NAMES

z is bound in P

free = no bound

Abelian Monoid law:

Commutativity $P|Q \equiv Q|P$

Associativity $P|(Q|R) \equiv (P|Q)|R$ — the same for +

Unit $P|0 \equiv P$

Unfolding Law: $K \equiv P$ if $K \triangleq P$

Scope extension: $(\nu x)0 \equiv 0$ $\quad(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$

if $u \overset{\neq}{=} v$ then $(\nu x)P \equiv (\nu x)(\text{if } u \overset{\neq}{=} v \text{ then } P)$ if $x \neq u \wedge x \neq v$

$P \mid (\nu x)Q \equiv (\nu x)(P|Q)$ if $x \notin fn(P)$

$+ \qquad\qquad +$

$\equiv$ is the smallest congruence satisfying the laws above.

implies

$$\underbrace{\underbrace{P \equiv P \qquad \frac{P \equiv Q}{Q \equiv P} \qquad \frac{P \equiv Q \quad Q \equiv R}{P \equiv R}}_{\text{equivalence}} \qquad \frac{P \equiv P'}{C[P] \equiv C[P']}}_{\text{Congruence}}$$

REDUCTION SEMANTICS

$$\tau . P + Q \longmapsto P$$

$$(a(x).P_1 + Q_1) \mid (\bar{a}u . P_2 + Q_2) \longmapsto P_1\{u/x\} \mid P_2$$

Substitution is a function $\sigma : N \to N$ (identity except for ~~finite~~ set of names)

$$\frac{P \longmapsto P'}{P \mid Q \longmapsto P' \mid Q}$$

$$\frac{P \longmapsto P'}{(\nu x) P \longmapsto (\nu x) P'}$$

$$\frac{P \equiv Q \quad Q \longmapsto Q' \quad Q' \equiv P'}{P \longmapsto P'}$$

$P\sigma$ is P where each free name x is replaced by $\sigma(x)$ with α-conv. if needed

LABELLED SEMANTICS

$$\alpha . P \xrightarrow{\alpha} P$$

$$\frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'}$$

$$\frac{P_1 \xrightarrow{\alpha} P_1' \quad bn(\alpha) \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\alpha} P_1' \mid P_2}$$

$$\frac{P_1 \xrightarrow{a(x)} P_1' \quad P_2 \xrightarrow{\bar{a}u} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1'\{u/x\} \mid P_2'}$$

Es.

$$\underbrace{a(x).P \mid Q}_{} \ \bar{a}u\, R$$

$\downarrow a(x)$

$P \mid Q$

$\downarrow \tau$

$(P \mid Q)\{u/x\} \mid R$

Q non deve avere x libero.

$$\frac{P \xrightarrow{\alpha} P' \quad x \notin \alpha}{(\nu x) P \xrightarrow{\alpha} (\nu x) P'}$$

$$\text{open} \quad \frac{P \xrightarrow{\bar{a}x} P' \quad a \neq x}{(\nu x) P \xrightarrow{\bar{a}\nu x} P'}$$

$$\frac{P \xrightarrow{\alpha} P'}{\text{if } x = x \text{ then } P \xrightarrow{\alpha} P'}$$

$$\frac{P \xrightarrow{\alpha} P' \quad \cancel{x \neq y}}{\text{if } x \neq y \text{ then } P \xrightarrow{\alpha} P'}$$

Example: home delivery pizza!

Customer $C \triangleq \overline{askPizza}\langle myHome \rangle . \overline{pay}\,\$. myHome(x) . \overline{eat}\,x$

Pizzaiolo $P \triangleq askPizza(y) . pay(z) . \overline{y}\,margherita . P$

Example: Electoral Propaganda (use of restriction)

$Speaker \triangleq \overline{air}\ VotaAntonio$

$Microphone \triangleq air(x) . \overline{wire}\,x$

$Loudspeaker \triangleq wire(y) . \overline{highvolume}\,y$

$ElectoralMeeting \triangleq Speaker \mid Microphone \mid Loudspeaker$

$\longmapsto$

$Rival \triangleq wire(z) . \overline{wire}\ VotaCiccio$ wire is public

$SecureElectoralMeeting \triangleq (\nu air)(\nu wire)(Speaker \mid Microphone \mid Loudspeaker)$

private channels

Example: different pizza any time (new names)

$Pizzaiolo \triangleq askPizza(y) . pay(z) . (\nu pizza)\, \overline{y}\,pizza . P$

$C \mid P \xrightarrow{\tau}{}^{*} (\nu pizza)(\overline{eat}\,pizza \mid P)$