

Déterminisme et confluence dans un π -calcul synchrone

Mehdi Dogguy

Preuves, Programmes et Systèmes



le 5 décembre 2007

Plan

- 1 Le π -calcul
- 2 Le $S\pi$ -calcul
- 3 Confluence
- 4 Typage

π quoi ?

Le π -calcul :

- a été inventé en 1991 par Robin Milner,
- est un langage de programmation théorique qui permet de raisonner sur les systèmes concurrents,
- manipule des processus.

Entités de base

$x, y, \dots \in \mathcal{X}$ ensemble de noms
 $P, Q, \dots \in \mathcal{P}$ processus

$P, Q ::= 0$ *processus mort*
 $| \overline{x} \langle y \rangle . P$ *envoi d'une valeur y sur x*
 $| x(y) . P$ *réception sur x*
 $| \nu x . P$ *création d'un nom*
 $| P + Q$ *choix non déterministe*
 $| P \mid Q$ *composition parallèle*
 $| !P$ *réplication : $!P = P \mid !P$*

Sémantique

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

$$\frac{j \in I}{\sum_{i \in I} \alpha_i.P_i \xrightarrow{\alpha_j} P_j}$$

$$\frac{P \xrightarrow{\bar{\alpha} \langle v \rangle} P' \quad Q \xrightarrow{\alpha(v)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

$$\frac{P \xrightarrow{\alpha} P'}{\nu x P \xrightarrow{\alpha} \nu x P'}$$

Exemple : Gestionnaire d'impression

$$\begin{aligned}
 \text{MANAGER} &= \nu \text{free} (\\
 &\quad \overline{\text{free}}\langle p_1 \rangle \mid \cdots \mid \overline{\text{free}}\langle p_n \rangle \\
 &\quad \mid !\text{HANDLER} \\
 &\quad) \\
 \text{HANDLER} &= \text{queue}(\text{task}, \text{callback}).\text{free}(p). \\
 &\quad \nu \text{cb} (\\
 &\quad \quad \overline{p}\langle \text{task}, \text{cb} \rangle \\
 &\quad \quad \mid \text{cb}().(\overline{\text{callback}}\langle \rangle \mid \overline{\text{free}}\langle p \rangle) \\
 &\quad)
 \end{aligned}$$

Traces

Ayant un ensemble d'action $\mathcal{L}$:

$$\alpha, \overline{\alpha}, \dots \in \mathcal{L}$$

On définit la trace d'un processus P comme :

$$Tr(P) = \{s \in \mathcal{L}^* \mid P \xRightarrow{s} \cdot\}$$

et on dit que les deux processus P et Q sont « **trace equivalent** » ssi $tr(P) = tr(Q)$.

Exemple : Distributeur de café (1)

```

1 coin().!(
    coffee().ccup<>.coin()
    + tea()    .tcup<>.coin()
  )

```

```

2 !(coin().(
    coffee().ccup<>
    + tea()    .tcup<>)
  )

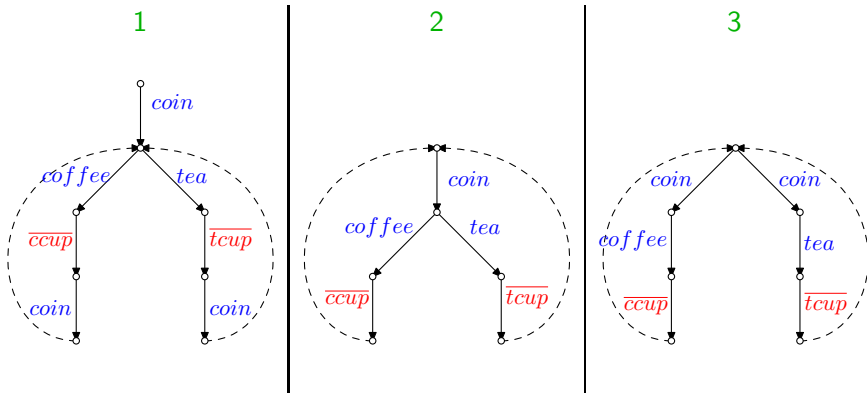
```

```

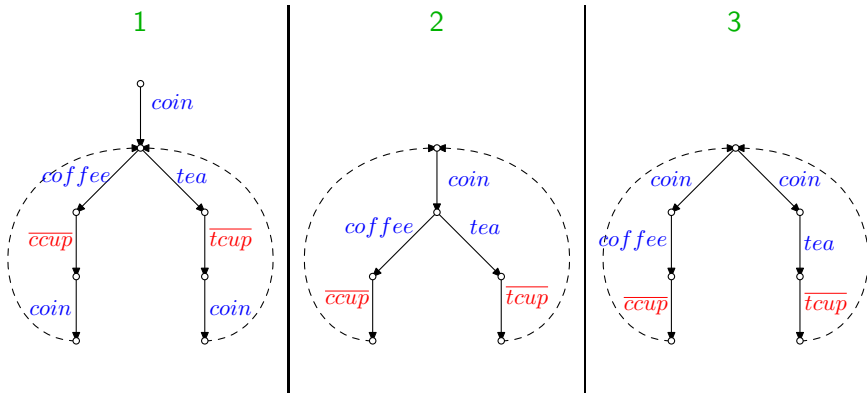
3 !(
    coin().coffee().ccup<>
    + coin().tea()    .tcup<>
  )

```

Exemple : Distributeur de café (2)



Exemple : Distributeur de café (2)



☠ $Tr(1) = Tr(2) = Tr(3)$ mais $1 \neq 2 \neq 3 \neq 1$

Simulation et Bisimulation

Definition

Une **simulation** est une relation binaire $\mathcal{R}$ sur l'ensemble des processus telle que pour tout P, Q, α :

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

Simulation et Bisimulation

Definition

Une **simulation** est une relation binaire $\mathcal{R}$ sur l'ensemble des processus telle que pour tout P, Q, α :

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

Definition

Une **bisimulation** est une relation binaire $\mathcal{R}$ sur les processus telle que $\mathcal{R}$ et $\mathcal{R}^{-1}$ sont des simulations.

P et Q sont bisimilaires ($P \approx Q$) s'il existe une bisimulation $\mathcal{R}$ telle que PRQ .

Plan

- 1 Le π -calcul
- 2 Le $S\pi$ -calcul
- 3 Confluence
- 4 Typage

- est un π -calcul **synchrone** :
 - les processus évoluent à la même vitesse,
 - et ils se synchronisent avec l'action spéciale N
(Chaque transition N marque la fin de l'instant courant).
- pas d'opérateur de choix non déterministe,
- la réception dépend de l'instant,
- opérateur de dérérérenciation $!s$,
- l'émission est persistente dans l'instant.

Exemples

$$pause.K = \nu s(\textcolor{green}{s}(\textcolor{blue}{s}(x)).0, K)$$

Exemples

$$pause.K = \nu s(\mathbf{s}(x).0, K)$$

$$A = \nu s, s', s''(\overline{s}\langle 1 \rangle \mid \overline{s}\langle 2 \rangle \mid s'(x).\overline{s''}\langle 1 \rangle, 0 \mid pause.\overline{s''}\langle !s \rangle)$$

Exemples

$$pause.K = \nu s(\mathbf{s}(x).0, K)$$

$$\begin{array}{c}
 A = \nu s, s', s''(\overline{s}\langle 1 \rangle \mid \overline{s}\langle 2 \rangle \mid \mathbf{s}'(x).\overline{s''}\langle 1 \rangle, 0 \mid pause.\overline{s''}\langle !s \rangle) \\
 \downarrow N \\
 A = \nu s, s', s''(0 \mid 0 \mid 0 \mid \overline{s''}\langle [1; 2] \rangle)
 \end{array}$$

Exemples

$$pause.K = \nu s(\mathbf{s}(x).0, K)$$

$$\begin{array}{c}
 A = \nu s, s', s''(\overline{s}\langle 1 \rangle \mid \overline{s}\langle 2 \rangle \mid \mathbf{s}'(x).\overline{s''}\langle 1 \rangle, 0 \mid pause.\overline{s''}\langle !s \rangle) \\
 \downarrow N \\
 A = \nu s, s', s''(0 \mid 0 \mid 0 \mid \overline{s''}\langle [2; 1] \rangle)
 \end{array}$$

Dans l'instant

$$(out) \quad \frac{e \text{ s'évalue en } v}{\overline{s}\langle e \rangle \xrightarrow{\overline{s}\langle v \rangle} \overline{s}\langle e \rangle}$$

$$(in) \quad \frac{}{s(x).P, K \xrightarrow{sv} [v/x]P \mid \overline{s}\langle v \rangle}$$

$$(comp) \quad \frac{P_1 \xrightarrow{\mu} P'_1 \quad bn(\mu) \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\mu} P'_1 \mid P_2}$$

$$(synch) \quad \frac{P_1 \xrightarrow{\nu t \overline{s}\langle v \rangle} P'_1 \quad P_2 \xrightarrow{sv} P'_2 \quad \{t\} \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\tau} \nu t(P'_1 \mid P'_2)}$$

$$(=1^{ind}) \quad \frac{match(v, p) = \theta}{[v \triangleright p]P_1, P_2 \xrightarrow{\tau} \theta P_1}$$

$$(=2^{ind}) \quad \frac{match(v, p) = \uparrow}{[v \triangleright p]P_1, P_2 \xrightarrow{\tau} P_2}$$

À la fin de l'instant

$$(0) \quad \frac{}{0 \xrightarrow{\emptyset, V} 0} \qquad (reset) \quad \frac{e \text{ s'évalue en } v \quad v \in V(s)}{\bar{s} \langle e \rangle \xrightarrow{[\{v\}/s], V} 0}$$

$$(cont) \quad \frac{s \notin \text{dom}(V)}{s(x).P, K \xrightarrow{\emptyset, V} V(K)} \qquad (par) \quad \frac{P_i \xrightarrow{E_i, V} P'_i \quad i = 1, 2}{(P_1 \mid P_2) \xrightarrow{E_1 \vee E_2, V} (P'_1 \mid P'_2)}$$

$$(next) \quad \frac{P \leq \nu s \, P' \quad P' \xrightarrow{E, V} P'' \quad V \Vdash E}{P \xrightarrow{N} \nu s \, P''}$$

Définition (une première bisimulation)

Une relation symétrique $\mathcal{R}$ sur les programmes est une bisimulation étiquetée quand $P \mathcal{R} Q$ implique :

- (L1) Si $P \xrightarrow{\tau} P'$, alors $\exists Q'$ tel que $(Q \xRightarrow{\tau} Q' \text{ et } P' \mathcal{R} Q')$.
- (L2) Si $P \xrightarrow{\nu \bar{s} \langle v \rangle} P'$, $\{\mathbf{t}\} \cap \text{fn}(Q) = \emptyset$, alors $\exists Q'$ tel que $(Q \xRightarrow{\nu \bar{s} \langle v \rangle} Q' \text{ et } P' \mathcal{R} Q')$.
- (L3) Si $P \xrightarrow{\text{sv}} P'$, alors $\exists Q'$ tel que $((Q \xRightarrow{\text{sv}} Q' \text{ et } P' \mathcal{R} Q') \text{ ou } (Q \xRightarrow{\tau} Q' \text{ et } P' \mathcal{R}(Q' \mid \bar{s} \langle v \rangle)))$.
- (L4) Si $S = \bar{s}_1 \langle v_1 \rangle \mid \dots \mid \bar{s}_n \langle v_n \rangle$, $n \geq 0$, $P' = (P \mid S) \downarrow$ et $P' \xrightarrow{\mathbf{N}} P''$ alors $\exists Q', Q''$ tels que $(Q \mid S) \xRightarrow{\tau} Q'$, $Q' \downarrow$, $P' \mathcal{R} Q'$, $Q' \xrightarrow{\mathbf{N}} Q''$, et $P'' \mathcal{R} Q''$.

On notera $\approx_1$ la plus large bisimulation étiquetée.

Définition (Bisimulation Standard)

Une relation symétrique $\mathcal{R}$ sur les programmes est une bisimulation étiquetée si $bn(\alpha) \cap fn(Q) = \emptyset$ et :

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

On notera $\approx$ la plus large bisimulation étiquetée.

$$(out) \quad \frac{e \text{ s'évalue en } v}{\overline{s}\langle e \rangle \xrightarrow{\overline{s}\langle v \rangle} \overline{s}\langle e \rangle}$$

$$(in_{aux}) \quad \frac{}{s(x).P, K \xrightarrow{s?v} [\nu/x]P}$$

$$(in) \quad \frac{}{P \xrightarrow{sv} (P \mid \overline{s}\langle v \rangle)}$$

$$(comp) \quad \frac{P_1 \xrightarrow{\mu} P'_1 \quad bn(\mu) \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\mu} P'_1 \mid P_2}$$

$$(synch) \quad \frac{P_1 \xrightarrow{\nu \overline{s}\langle v \rangle} P'_1 \quad P_2 \xrightarrow{s?v} P'_2 \quad \{t\} \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\tau} \nu t(P'_1 \mid P'_2)}$$

$$(=1^{ind}) \quad \frac{match(v, p) = \theta}{[\nu \triangleright p]P_1, P_2 \xrightarrow{\tau} \theta P_1}$$

$$(=2^{ind}) \quad \frac{match(v, p) = \uparrow}{[\nu \triangleright p]P_1, P_2 \xrightarrow{\tau} P_2}$$

Théorème

$\approx_1$ et $\approx$ coïncident.

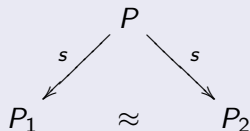
Plan

- 1 Le π -calcul
- 2 Le $S\pi$ -calcul
- 3 Confluence**
- 4 Typage

Déterminisme

Définition

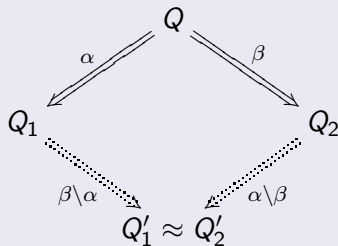
P est déterministe si pour n'importe quelle séquence d'action s :



Confluence

Définition

P est **confluent** si pour chaque dérivé Q , on a $\alpha \downarrow \beta$ et :



Théorème

P est déterministe $\Leftrightarrow P$ est confluent.

Plan

- 1 Le π -calcul
- 2 Le $S\pi$ -calcul
- 3 Confluence
- 4 Typage**

Deux situations à traiter :

- Au moins deux valeurs différentes sont émises sur le même signal dans le même instant :

$$\overline{s}\langle v \rangle \mid \overline{s}\langle v' \rangle \mid s(x).P, K$$

- À la fin de l'instant, au moins deux valeurs sont disponibles pour réception :

$$\overline{s}\langle v \rangle \mid \overline{s}\langle v' \rangle \mid \text{pause}.A(!s)$$

$$s : \text{Sig}^{\omega}_{\binom{n}{m}{p}}(\sigma)$$

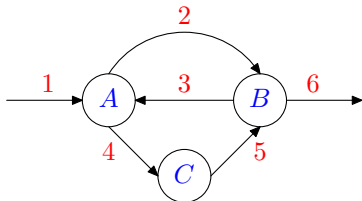
où : $\begin{cases} n & \text{le nombre maximum d'émissions} \\ m & \text{le nombre maximum de réceptions} \\ p & \text{le nombre maximum de réceptions, à la fin de l'instant} \end{cases}$

$$n, m, p \in L = \{0 < 1 < \infty\}$$

Flot d'information Synchrone (Synchronous Data Flow)

$$\nu \mathbf{s_2, s_3, s_4, s_5} (A(s_1, s_2, s_3, s_4) \mid B(s_2, s_3, s_5, s_6) \mid C(s_4, s_5))$$

$$\begin{aligned} A(s_1, s_2, s_3, s_4) &= \mathbf{s_1}(x).(\overline{\mathbf{s_2}}\langle f(x) \rangle \mid \mathbf{s_3}(y).(\overline{\mathbf{s_4}}\langle g(y) \rangle \mid \text{pause}.A(s_1, s_2, s_3, s_4)), 0), 0 \\ B(s_2, s_3, s_5, s_6) &= \mathbf{s_2}(x).(\overline{\mathbf{s_3}}\langle i(x) \rangle \mid \mathbf{s_5}(y).(\overline{\mathbf{s_6}}\langle l(y) \rangle \mid \text{pause}.B(s_2, s_3, s_5, s_6)), 0), 0 \\ C(s_4, s_5) &= \mathbf{s_4}(x).(\overline{\mathbf{s_5}}\langle h(x) \rangle \mid \text{pause}.C(s_4, s_5)), 0 \end{aligned}$$

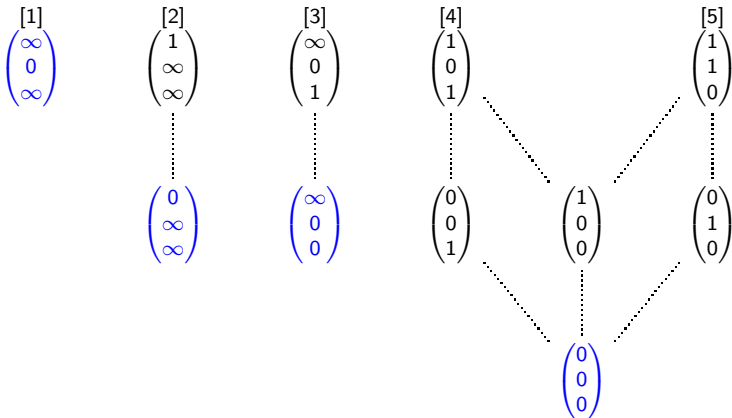


Client-Serveur

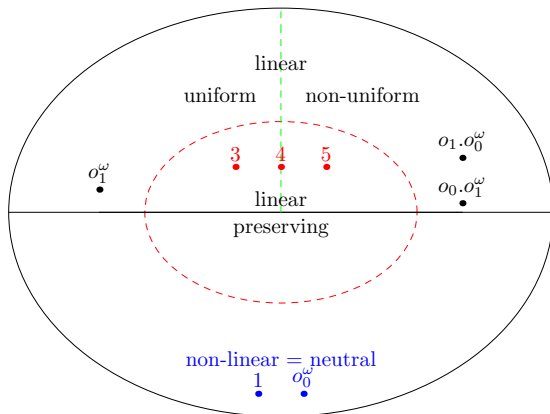
$$\begin{aligned}
 \text{Server}(s) &= \text{pause.Handle}(s, !s) \\
 \text{Handle}(s, l) &= [l \triangleright \text{cons}(\text{req}(s', x), l')] \\
 &\quad (\overline{s'} \langle f(x) \rangle \mid \text{Handle}(s, l')), \\
 &\quad \text{Server}(s)
 \end{aligned}$$

$$\begin{aligned}
 \text{Client}(x, s, t) &= \nu s' (\\
 &\quad \overline{s} \langle \text{req}(s', x) \rangle \\
 &\quad \mid \text{pause}.s'(x).\overline{t} \langle x \rangle, 0 \\
 &\quad)
 \end{aligned}$$

Les 5 familles de signaux



Classification des signaux



$$\sigma_{\kappa} ::= 1 \mid List_{\infty}(\sigma_{\kappa}) \mid Set_{\infty}(\sigma_{\kappa}) \mid Sig_u(\sigma_{\kappa})$$

$$\sigma_{\lambda} ::= List_1(\sigma_{\kappa\lambda}) \mid Set_1(\sigma_{\kappa\lambda}) \mid Sig_u(\sigma_{\kappa}) \mid Sig_v(\sigma_{\lambda})$$

$$\sigma_{\kappa\lambda} ::= \sigma_{\kappa} \mid \sigma_{\lambda}$$

$$\rho ::= \sigma_{\kappa\lambda} \mid Sig_u(\sigma_{\kappa}) \mid Sig_v(\sigma_{\kappa\lambda})$$

Addition sur les types

$$\begin{array}{rclcl}
 0 & \oplus & x & = & x \\
 1 & \oplus & 1 & = & \uparrow \\
 \infty & \oplus & x & = & \infty
 \end{array}$$

$$u \oplus u' \text{ is defined} \quad \Leftrightarrow \quad [u, u' \in \text{Class}(n) \wedge u \oplus u' \in \text{Class}(n)]$$

$$\begin{array}{rclcl}
 \sigma & \oplus & \sigma & = & \sigma \\
 Op_{\infty}(\sigma_1, \dots, \sigma_n) & \oplus & Op_{\infty}(\sigma_1, \dots, \sigma_n) & = & Op_{\infty}(\sigma_1, \dots, \sigma_n) \\
 Sig_u(\sigma) & \oplus & Sig_v(\sigma) & = & Sig_{u \oplus v}(\sigma) \\
 \rho & \oplus & \rho' & = & \uparrow
 \end{array}$$

Typage des expressions

$$(var) \quad \frac{\Gamma \text{ neutral}}{\Gamma, x : \rho \vdash x : \rho}$$

$$(k) \quad \frac{\begin{array}{l} \Gamma_i \vdash r_i : \sigma_i \quad i = 1, \dots, n \quad \Gamma_0 \text{ neutral} \\ k : (\sigma_1, \dots, \sigma_n) \rightarrow \sigma \quad k = f \text{ or } k = c \end{array}}{\Gamma_0 \oplus \Gamma_1 \oplus \dots \oplus \Gamma_n \vdash k(r_1, \dots, r_n) : \sigma}$$

$$(!_{Set_i}) \quad \frac{\begin{array}{l} \Gamma \text{ neutral} \\ u \in U_n(1, 3) \quad i = \pi_3(u[0]) \end{array}}{\Gamma, s : Sig_u(\sigma) \vdash !s : Set_i(\sigma)}$$

$$(!_{List_i}) \quad \frac{\begin{array}{l} \Gamma \text{ neutral} \\ u \in U_n(2, 4) \quad i = \pi_3(u[0]) \end{array}}{\Gamma, s : Sig_u(\sigma) \vdash !s : List_i(\sigma)}$$

Typage des programmes

$$(0) \quad \frac{\Gamma \text{ neutral}}{\Gamma \vdash 0}$$

$$(out) \quad \frac{u \in U_o \quad \Gamma_1 \vdash s : \text{Sig}_u(\sigma) \quad \Gamma_2 \vdash e : \sigma}{\Gamma_1 \oplus \Gamma_2 \vdash \bar{s}\langle e \rangle}$$

$$(\nu) \quad \frac{\Gamma, s : \text{Sig}_u(\sigma) \vdash P}{\Gamma \vdash \nu s P}$$

$$(in) \quad \frac{u \in U_l \quad \Gamma_1 \vdash s : \text{Sig}_u(\sigma) \quad \Gamma_2, x : \sigma \vdash P \quad \uparrow(\Gamma_1 \oplus \Gamma_2) \vdash K}{\Gamma_1 \oplus \Gamma_2 \vdash \mathbf{s}(x).P, K}$$

$$(par) \quad \frac{\Gamma_i \vdash P_i \quad i = 1, 2}{\Gamma_1 \oplus \Gamma_2 \vdash P_1 \mid P_2}$$

$$(rec) \quad \frac{A : (\rho_1, \dots, \rho_n) \quad \Gamma_i \vdash r_i : \rho_i \quad i = 1, \dots, n}{\Gamma_1 \oplus \dots \oplus \Gamma_n \vdash A(r_1, \dots, r_n)}$$

$$(m_s) \quad \frac{\Gamma \vdash P_i \quad i = 1, 2}{\Gamma \vdash [s_1 = s_2]P_1, P_2}$$

$$(m_c) \quad \frac{c : (\sigma_1, \dots, \sigma_n) \rightarrow \sigma \quad \Gamma' \vdash u : \sigma \quad \Gamma, x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash P_1 \quad \Gamma \oplus \Gamma' \vdash P_2}{\Gamma \oplus \Gamma' \vdash [\mathbf{u} \sqsubseteq c(x_1, \dots, x_n)]P_1, P_2}$$

Proposition (Préservation du typage par réduction)

Supposons que $\Gamma \vdash P$. Alors :

- (1) Si $P \xrightarrow{sv} P'$, $\Gamma' \vdash \overline{s} \langle v \rangle$ et $\Gamma \oplus \Gamma'$ est défini, alors $\Gamma \oplus \Gamma' \vdash P'$.
- (2) Si $P \xrightarrow{\nu t : \rho \overline{s} \langle v \rangle} P'$ alors $\Gamma, t : \rho \vdash P'$.
- (3) Si $P \xrightarrow{\tau} P'$ alors $\exists \Gamma'$ tel que $\Gamma' \vdash P'$.
- (4) Si $P \xrightarrow{N} P'$ alors $\uparrow(\Gamma) \vdash P'$.

Théorème (Typage $\Rightarrow$ déterminisme)

Si le programme P est typable dans un contexte Γ , alors P est déterministe.

Merci ...