Laboratoire d'Informatique
de l'École Polytechnique

Conception de π-*sa* : un analyseur statique générique pour le π-calcul

Jérôme Feret

sous la direction de :

Radhia Cousot
Arnaud Venet

le 14 septembre 1999

π-calcul : Généralités

Le π-calcul est un formalisme servant à représenter des réseaux de processus qui peuvent communiquer entre eux par l'intermédiaire de canaux.

Le π-calcul est plus expressif que CCS

- la topologie du réseau est dynamiquement modifiée lors des communications (code mobile)
- certains processus peuvent se dupliquer un nombre arbitraire de fois,

  $\Longrightarrow$ chaque instance récursive déclare ses propres canaux, qu'elle peut ensuite exporter vers les autres processus.

$\Longrightarrow$ On ne peut connaître à l'avance la topologie du réseau.

Principe du stage

La sémantique du π-calcul est donnée par une relation de transition $\rightarrow$.

Approximer la sémantique collectrice d'un processus,

$$S(P_0) = \{P' | P_0 \rightarrow^* P'\} = lfp_\perp F$$

$$\text{où } F : X \mapsto \{P_0\} \cup \{P' | \exists P \in X,\ P \rightarrow P'\}$$

en utilisant une sémantique abstraite donnée par

- un domaine abstrait $\mathfrak{D}^\sharp$
- une fonction de concrétisation γ
- une fonction abstraite $F^\sharp$

vérifiant la contrainte de cohérences :

$$\forall d^\sharp \in \mathfrak{D}^\sharp,\ [F \circ \gamma](d^\sharp) \subseteq [\gamma \circ F^\sharp](d^\sharp)$$

Théorème 1 Sous ces conditions :

$$S(P_0) \subseteq lfp_\perp(F^\sharp)$$

Syntaxe du π-calcul

Soit *Channel* un ensemble infini dénombrable de noms de canaux.

$$
\begin{array}{rcll}
P & ::= & \text{action}.P & \text{(action)} \\
 & | & (P \mid P) & \text{(exécution concurrente)} \\
 & | & (P + P) & \text{(exécution disjonctive)} \\
 & | & *P & \text{(réplication)} \\
 & | & \emptyset & \text{(fin d'un processus)}
\end{array}
$$

$$
\begin{array}{rcll}
\text{action} & ::= & c![x_1, ..., x_n] & \text{(émission d'un } \textit{message}\text{)} \\
 & | & c?[x_1, ..., x_n] & \text{(attente d'un } \textit{message}\text{)} \\
 & | & (\nu x) & \text{(création d'un canal)}
\end{array}
$$

où c, x_1, ..., x_n, x, $y \in$ *Channel*

$\Longrightarrow$ ν et ? sont les seuls lieurs de canaux.

Applications

Le π-calcul peut représenter par exemple :

- des réseaux téléphoniques (ERLANG)
- des protocoles d'échange de clefs
- l'interaction entre les processus d'un programme parallèle réel en faisant abstraction des aspects opérationnels de celui-ci.

Exemple : un serveur 3 ports

$$
\begin{aligned}
&\text{Allouer} := \\
&\quad *\text{make}?[] \\
&\qquad (\nu\, \textit{canal-entrée})(\nu\, \textit{canal-sortie})(\nu\, \textit{info-client}) \\
&\qquad\quad (\ \textit{canal-entrée}![\textit{info-client}] \\
&\qquad\quad \mid\ \textit{canal-entrée}?[\textit{info-reçue}]. \\
&\qquad\qquad\qquad (\textit{canal-sortie}![\textit{info-reçue}] \mid \text{make}![])) \\
&\text{Port} := \\
&\quad \text{make}![] \\
&\text{Serveur} := \text{Allouer} \mid \text{Port} \mid \text{Port} \mid \text{Port}
\end{aligned}
$$

Relation de transition

La relation de transition spécifie comment les processus s'exécutent.

$$
\begin{array}{rcll}
P + Q & \rightarrow & P & \text{(choix gauche)} \\
P + Q & \rightarrow & Q & \text{(choix droit)} \\
c![\overline{x}]P \mid c?[\overline{y}]Q & \rightarrow & P \mid Q[\overline{y} \leftarrow \overline{x}] & \text{(communication)}
\end{array}
$$

$$\frac{P \rightarrow Q}{(\nu x)P \rightarrow (\nu x)Q}$$

$$\frac{P' \equiv P \; P \rightarrow Q \; Q \equiv Q'}{P' \rightarrow Q'}$$

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

Une relation de congruence permet de modifier la syntaxe d'un processus

$\Longrightarrow$ Révéler les rédex

Une exécution possible pour le serveur

$$
\begin{array}{l}
(*\mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\ \mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid\ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid\ \mathrm{make}![] \mid\ \mathrm{make}![] \mid\ \mathrm{make}![]) \\
\equiv \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad *P \equiv *P \mid P\ \text{(réplication)} \\
(*\mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\ \mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid\ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid\ \mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid\ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid\ \mathrm{make}![] \mid\ \mathrm{make}![] \mid\ \mathrm{make}![])
\end{array}
$$

$$
\begin{array}{l}
(*\mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ \mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ \mathrm{make}![] \mid \mathrm{make}![] \mid \mathrm{make}![])
\end{array}
$$

$$\rightarrow \text{(communication)}$$

$$
\begin{array}{l}
(*\mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ \mathrm{make}![] \mid \mathrm{make}![])
\end{array}
$$

$$
\begin{array}{l}
(*\mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ \mathrm{make}![] \mid \mathrm{make}![])
\end{array}
$$

$\rightarrow$ (communication)

$$
\begin{array}{l}
(*\mathrm{make}?[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}![\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}reçue}] \mid \mathrm{make}![])) \\
\mid \ (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}sortie}![\mathit{info\text{-}client}] \mid \mathrm{make}![]) \\
\mid \ \mathrm{make}![] \mid \mathrm{make}![])
\end{array}
$$

Analyse

On veut utiliser le π-calcul pour montrer des propriétés sur les réseaux ou les protocoles qu'il représente :

- ressources physiques :

  En exprimant, le nombre d'instances des sous-processus présents simultanément en cours d'exécution, on obtient une information sur la quantité de ressource physique nécessaire pour simuler ce processus.

  $\Longrightarrow$ Montrer que jamais plus de trois clients sont en attente.

- sécurité d'un protocole :

  La sécurité d'un protocole s'exprime sous forme de contraintes sur la topologie des communications du processus.

  $\Longrightarrow$ Vérifier que l'information traitée par le serveur est retournée au bon client.

Une sémantique inadéquate

La sémantique ne permet pas d'exprimer de telles propriétés :

- La règle de réplication permet de dupliquer les ressources un nombre arbitraire de fois.

  $\Longrightarrow$ Il est alors superflu de compter les processus

- La règle d'α-conversion permet de renommer les canaux

  $\Longrightarrow$ Il est impossible d'établir des invariants sur les noms de canaux.

Sémantique paresseuse de Turner

$\Longrightarrow$ Maîtriser les réplications de ressource

Pour cela, on opère quelques modifications sur :

- la syntaxe :

  Tout signe de réplication doit être suivi d'une attente de *message*.

- la sémantique opérationnelle :

  On remplace la règle de congruence (réplication) par une règle de transition :

$c![\overline{x}]P|*c?[\overline{y}]Q \rightarrow P|*c?[\overline{y}]Q|Q[\overline{y} \leftarrow \overline{x}]$ (réplication)

Sémantique non-standard

On manipule des ensembles de sous-processus munis :

- d'un identifiant
  $\Longrightarrow$ distinguer les instances récursives
- d'un environnement
  $\Longrightarrow$ spécifier l'origine de ses canaux

$$
\begin{array}{l}
(\nu \mathrm{make}) \\
\quad (*\mathrm{make}?^{0}[] \\
\qquad\qquad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad\qquad\quad (\ \mathit{canal\text{-}entrée}!^{1}[\mathit{info\text{-}client}] \\
\qquad\qquad\quad \mid \mathit{canal\text{-}entrée}?^{2}[\mathit{info\text{-}reçue}]. \\
\qquad\qquad\qquad\qquad (\mathit{canal\text{-}sortie}!^{3}[\mathit{info\text{-}reçue}] \mid \mathrm{make}!^{4}[])) \\
\quad \mid \mathrm{make}!^{6}[] \mid \mathrm{make}!^{7}[] \mid \mathrm{make}!^{8}[])
\end{array}
$$

$$
\left\{
\begin{array}{l}
\left(0, \varepsilon, \left\{ \mathrm{make} \mapsto (\mathrm{make}, \varepsilon) \quad \right.\right) \\
\left(3, (1,6), \left\{
\begin{array}{ll}
\mathit{canal\text{-}sortie} & \mapsto (\mathit{canal\text{-}sortie}, (1,6)) \\
\mathit{info\text{-}reçue} & \mapsto (\mathit{info\text{-}client}, (1,6))
\end{array}
\right.\right) \\
\left(3, (1,7), \left\{
\begin{array}{ll}
\mathit{canal\text{-}sortie} & \mapsto (\mathit{canal\text{-}sortie}, (1,7)) \\
\mathit{info\text{-}reçue} & \mapsto (\mathit{info\text{-}client}, (1,7))
\end{array}
\right.\right) \\
\left(4, (1,6), \left\{ \mathrm{make} \mapsto (\mathrm{make}, \varepsilon) \right.\right) \\
\left(4, (1,7), \left\{ \mathrm{make} \mapsto (\mathrm{make}, \varepsilon) \right.\right) \\
\left(8, \varepsilon, \left\{ \mathrm{make} \mapsto (\mathrm{make}, \varepsilon) \right.\right)
\end{array}
\right\}
$$

Cohérence

Théorème 2 La sémantique standard et la sémantique non-standard sont en *bisimulation faible*.

Le point essentiel de la démonstration est de montrer qu'il n'y a jamais de conflits entre les noms de canaux.

Sémantique abstraite

On spécifie une approximation de la sémantique collectrice.

Cette sémantique abstraite dépend de deux domaines génériques :

- $Id^\sharp_1$ et $Id^\sharp_2$

  pour représenter respectivement des identifiants et des couples d'identifiants.

- $\mathfrak{V}$

  pour compter les instances textuelles de chaque sous-processus

et d'une fonction de partitionnement f.

La sémantique abstraite est donnée sous la forme d'une relation de transition qui mime la relation de transition concrète.

Exclusion mutuelle

$$
\begin{aligned}
A &:= *a?^{1}[x](x!^{2}[a] + c?^{3}[u]\cancel{d!^{4}[u]}) \\
B &:= *b?^{5}[x](x!^{6}[b] + c!^{7}[e]\ ) \\
C &:= a!^{8}[b] \\
P &:= A \mid B \mid C
\end{aligned}
$$

$\Longrightarrow$ Les sous-processus 3 et 7 ne peuvent pas communiquer

car le système

$$
\begin{cases}
\sharp(2) + \sharp(3) + \sharp(6) + \sharp(7) + \sharp(8) = 1 \\
\sharp(3) \in [|1; \infty|[\\
\sharp(7) \in [|1; \infty|[
\end{cases}
$$

est insatisfiable dans $\mathbb{N}^{+}$.

Le serveur 3 ports

$$
\begin{array}{l}
(*\text{make}?^{1}[] \\
\quad (\nu \mathit{canal\text{-}sortie})(\nu \mathit{info\text{-}client}) \\
\qquad (\text{canal-entrée}!^{2}[\mathit{info\text{-}client}, \mathit{canal\text{-}sortie}] \\
\qquad \mid \text{canal-entrée}?^{3}[\mathit{info\text{-}reçue}]. \\
\qquad\qquad (\mathit{canal\text{-}sortie}!^{4}[\mathit{info\text{-}reçue}] \mid \text{make}!^{5}[])) \\
\mid \text{make}!^{6}[] \mid \text{make}!^{7}[] \mid \text{make}!^{8}[] \\
\mid \text{make}?^{9}[]\text{make}?^{10}[]\text{make}?^{11}[]\text{make}?^{12}[]\cancel{\text{test}!^{13}[]} \\
)
\end{array}
$$

$$
\begin{cases}
\sharp(1) = 1 \\
\sharp(13) = 0 \\
\sharp(4) \in [|0; \infty|[\\
\sharp(i) \in [|0; 3|], \ \forall i \in \{2; 3; 5\} \\
\sharp(i) \in [|0; 1|], \ \forall i \in [|6; 12|] \\
\displaystyle\sum_{i \in [|5;8|]} \sharp(i) = 3 - \sharp(2) - \sharp(10) - 2 \times \sharp(11) - 3 \times \sharp(12)
\end{cases}
$$

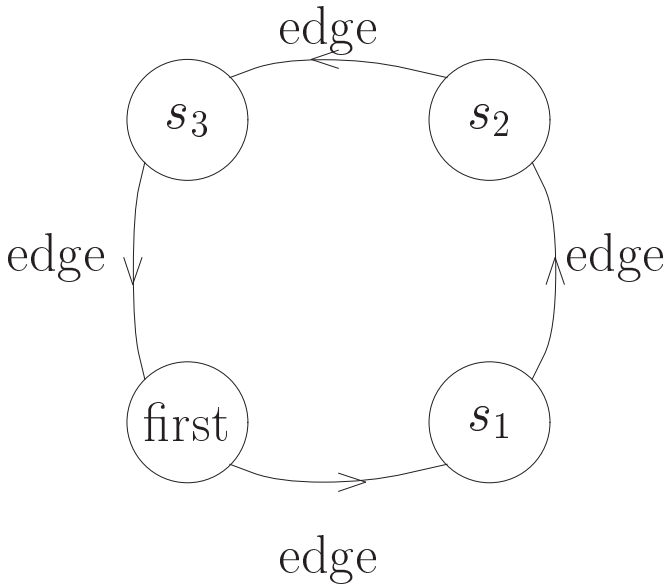
Le serveur erroné

$$
\begin{array}{l}
(*\text{make}?^{1}[] \\
\quad (\nu\, \mathit{canal\text{-}entrée})(\nu\, \mathit{canal\text{-}sortie})(\nu\, \mathit{info\text{-}client}) \\
\qquad (\mathit{canal\text{-}entrée}!^{2}[\mathit{info\text{-}client}] \\
\qquad \mid \ \mathit{canal\text{-}entrée}?^{3}[\mathit{info\text{-}reçue}].\ \mathit{canal\text{-}sortie}!^{4}[\mathit{info\text{-}reçue}] \\
\qquad \mid \ \text{make}!^{5}[]) \\
\mid \ \text{make}!^{6}[] \ \mid \ \text{make}!^{7}[] \ \mid \ \text{make}!^{8}[])
\end{array}
$$

$$
\begin{cases}
\sharp(1) = 1 \\
\sharp(13) \in [|0;1|] \\
\sharp(i) \in [|0;\infty|[,\ \forall i \in \{2;3;4;5\} \\
\sharp i \in [|0;1|],\ \forall i \in [|6;12|] \\
\sharp(3) + \sharp(4) = \displaystyle\sum_{i \in \{5,6,7,8\}} \underline{\sharp}(1,i)
\end{cases}
$$

L'anneau de processus

$$
\begin{array}{l}
(\nu \mathrm{make})(\nu \mathrm{edge})(\nu \mathrm{first}) \\
\quad (*\mathrm{make}?^{1}[\mathit{last}](\nu \mathit{next}) \\
\qquad\qquad\qquad\qquad (\mathrm{edge}!^{2}[\mathit{last},\mathit{next}] \\
\qquad\qquad\qquad\qquad \mid\ \mathrm{make}!^{3}[\mathit{next}]) \\
\quad \mid\ *\mathrm{make}?^{6}[\mathit{last}](\mathrm{edge}!^{7}[\mathit{last},\mathrm{first}]) \\
\quad \mid\ \mathrm{make}!^{8}[\mathrm{first}])
\end{array}
$$



$$\sharp\!\!\sharp(1,3) + 1 = \sharp\!\!\sharp(1,3)$$

Les problèmes d'échappement

$$
\begin{array}{l}
(\nu \mathrm{make})(\nu \mathrm{test}) \\
\quad (*\mathrm{make}?^{1}[](\nu\, a)(\nu\, b) \\
\qquad\qquad\qquad (\ a?^{2}[]\, b?^{3}[]\cancel{\mathrm{test}!^{4}[]} \\
\qquad\qquad\qquad \mid\ a?^{5}[]\, b!^{6}[] \\
\qquad\qquad\qquad \mid\ a!^{7}[] \\
\qquad\qquad\qquad \mid \mathrm{make}!^{8}) \\
\quad \mid \mathrm{make}!^{9})
\end{array}
$$

$\Longrightarrow$ Analyse un sous-processus indépendamment de son contexte.

- un ensemble de canaux publics
- le contexte peut écouter et envoyer des *messages* sur les canaux publics.

Conclusion

On a ainsi développé une analyse précise capable de :

- détecter des exclusions mutuelles
- distinguer deux instances d'un processus récursif
- analyser un sous-processus indépendamment de son contexte

Perspectives

- améliorer cette analyse
  - concevoir un domaine plus précis pour représenter les environnements
  - développer une analyse modulaire
- adapter cette analyse à d'autres formalismes
  - *les Ambiants*
  - un langage réel (ERLANG)