HOCore en Coq

Aurèle Barrière, sur un article de Petar Maskimovic & Alan Schmitt

10 mai 2016

Des calculs

λ-calcul

$$\Lambda ::= x \mid \Lambda\Lambda \mid \lambda x.\Lambda$$

x variable, Λ fonction.

π-calcul

$$P ::= a(n).P \mid \bar{a}\langle n\rangle.P \mid P\|P \mid 0 \mid \nu n P \mid !P$$

a canal, n nom, P processus.

Calcul de processus

- Turing Complet
- Minimal
- Ordre supérieur

Higher **O**rder **Core**

Syntaxe

Catégories syntaxiques

Variables x
Canaux a
Processus P

La grammaire d'un processus en HOCore

$$P ::= a(x).P \mid \bar{a}\langle P \rangle \mid P \| P \mid x \mid 0$$

Sémantique

$$P ::= a(x).P \mid \bar{a}\langle P \rangle \mid P \| P \mid x \mid 0$$

0 ne fait rien.
x variable.
$P \| Q$ exécution en parallèle. Associative et commutative. Permet communication
$\bar{a}\langle P \rangle$ émission de P sur le canal a.
$a(x).P$ réception sur le canal a pour x dans P.

Simplifications

Émission et réception sur un même canal.

$$\bar{a}\langle P\rangle \| a(x).Q \to [P/x]Q$$

$\to$: réduction.
Parallèlisme associatif et commutatif donc

$$\bar{a}\langle P\rangle \| \bar{b}\langle Q\rangle \| a(x).x \equiv \bar{b}\langle Q\rangle \| \bar{a}\langle P\rangle \| a(x).x \to \bar{b}\langle Q\rangle \| P$$

Exemple : récursivité

Processus P. On cherche $!P$ tel que $!P \rightarrow P \| !P$.
Alors $!P$ va répliquer indéfiniment P.
Soit $L = r(x).(x \| \overline{r}\langle x \rangle)$.
Soit $R = \overline{r}\langle P \| r(x).(x \| \overline{r}\langle x \rangle) \rangle$
Montrons que $!P = L \| R$ convient.
L et R communiquent sur r. Après communication, R se réduit donc en 0.
Dans L, on remplace x par le message émis par R.
On a donc

$$!P \rightarrow P \| r(x).(x \| \overline{r}\langle x \rangle) \parallel \overline{r}\langle P \| r(x).(x \| \overline{r}\langle x \rangle) \rangle \| 0$$

Et donc $!P \rightarrow P \| !P$

$$!P = (r(x).(x \| \overline{r}\langle x \rangle)) \parallel \overline{r}\langle P \| r(x).(x \| \overline{r}\langle x \rangle) \rangle$$

Exemple : choix de processus

Choisir entre P et Q.
On souhaite donc avoir le processus $(a_1.P \oplus a_2.Q)$ et les processus $\hat{a}_1$ et $\hat{a}_2$ tels que

$$(a_1.P \oplus a_2.Q) \| \hat{a}_1 \rightarrow^* P$$

$$(a_1.P \oplus a_2.Q) \| \hat{a}_2 \rightarrow^* Q$$

$$(a_1.P \oplus a_2.Q) = \bar{a}_1\langle P \rangle \parallel \bar{a}_2\langle Q \rangle$$

$$\hat{a}_1 = a_1(X).(a_2(Y).(X))$$

$$\hat{a}_2 = a_1(X).(a_2(Y).(Y))$$

Complétude

On a récursivité, ordre supérieur, conditions...
HOCore est Turing-Complet.
Encodage dans les machines de Minsky.

Équivalence de processus

En HOCore, l'équivalence de processus est décidable.

Équivalence très faible.
P et Q équivalents ssi

- Si P se réduit en P', il existe Q' équivalent à P' tel que Q se réduit en Q'.
- P et Q ont les mêmes *observables* : ils émettent des messages sur les mêmes canaux.
- Pour tout contexte C (processus avec un trou), le contexte complété avec P, $C[P]$, est équivalent à $C[Q]$.

Formalisation en Coq

Intérêt de la formalisation : trouver une méthode pour décider de l'équivalence.
Syntaxe : OK
Sémantique : alpha-conversion

Alpha Conversion

Variables liées qui jouent le même rôle.
Exemple : $a(x).a(y).x$ et $a(z).a(y).z$ pour x et z.
Mais pas les processus x et y (libres)

Représentation canonique des noms

Idée : fonction qui calcule à chaque variable un indice indépendament du nom.
Pollack, Sato et Riciotti, *A Canonical Locally Named Representation of Binding*

Systèmes de transitions étiquetées

Analyser comportement processus : trouver les réductions. Mais deux processus qui communiquent en parallèle ne sont pas toujours à côté :

$$\bar{a}\langle P\rangle \| R \| S \| T \| U \| a(x).Q \text{ se réduit en } 0 \| R \| S \| T \| U \| [P/x]Q$$

On étiquette le comportement de chaque processus pour trouver les réductions.

Règles de LTS

$$\text{OUT } \bar{a}\langle P\rangle \xrightarrow{\bar{a}\langle P\rangle} 0$$

$$\text{IN } a(x).Q \xrightarrow{a(P)} [P/x]Q$$

$$\text{TAU1 Si } P \xrightarrow{\bar{a}\langle R\rangle} P' \text{ et } Q \xrightarrow{a(R)} Q' \text{ alors } P \| Q \xrightarrow{\tau} P' \| Q'$$

Correction de preuves

Formalisation en Coq : preuves assistées.
De nombreuses preuves sur le calcul contenaient des erreurs : hypothèses fausses, raisonnement sur de mauvaises structures...
Formalisation : 4kloc
Preuves : 22kloc

Conclusion

Formaliser la syntaxe, puis la sémantique en permettant à l'assistant de preuve de faire des réductions.
Résoudre les problèmes (alpha-conversion) et introduire de nouveaux outils (LTS) pour aboutir à de nouvelles méthodes de preuves.
Méthode pour décider l'équivalence de processus.
Corriger les preuves existantes, en apprendre plus sur ce type de calculs.
Première formalisation d'un π-calcul d'ordre supérieur.
Il reste des preuves à traduire en Coq.