De la KAM avec un Processus d'Ordre Supérieur

Damien Pous, Alan Schmitt

De la KAM avec un Processus d'Ordre Suprieur

D. Pous[1] & A. Schmitt[2]

1: CNRS, damien.pous@ens-lyon.fr
2: Inria, alan.schmitt@inria.fr

Résumé

Nous prsentons un encodage simple et direct de la machine abstraite de Krivine (KAM) dans le calcul de processus d'ordre suprieur HOcore, en utilisant un nombre trs restreint de canaux de communication. Cet encodage montre qu'il est possible de capturer l'expressivit du λ-calcul en HOcore ds que l'on fixe l'ordre d'valuation. Nous donnons galement une nouvelle borne infrieure pour le nombre minimal de restrictions ncessaire pour rendre l'quivalence de programmes dans HOcore indcidable. [1]

1. Introduction

Le calcul de processus HOcore est remarquable par sa similarit au λ-calcul, auquel il ajoute une notion de concurrence. Malgr cette similarit, aucun encodage du λ-calcul en HOcore n'a t prsent ce jour. En effet, l'appariement entre une fonction et son argument est trs syntaxique et trs rigide en λ-calcul, alors qu'il est beaucoup plus lche en HOcore car il correspond la prsence simultane sur le mme nom de canal d'un message et d'un rcepteur pour ce message. Cette diffrence cruciale, couple l'impossibilit de gnrer de nouveaux noms de canaux, implique que toute traduction doit fixer le nombre de redex pouvant tre activs en parallle. Ce nombre pouvant tre non-born pour certains λ-termes, ceci empche toute traduction *tant que la stratgie d'valuation n'a pas t fixe*. C'est cette deuxime voie que nous explorons ici, en choisissant une stratgie en appel par nom, dcline sous la forme d'une machine abstraite de Krivine (KAM).

Une fois ce choix de conception arrt, la traduction est trs naturelle : on reprsente la structure rcursive de la pile de la KAM comme deux messages transportant respectivement le terme de tte et, rcursivement, la queue de la pile. La β-rduction est simplement la communication entre le terme actif, reprsentant la fonction, avec la tte de la pile. De manire surprenante, cette traduction permet galement de capturer l'oprateur de contrle call-cc de la KAM. La rification de la continuation en tant que pile contenue dans un message permet en effet de facilement la dupliquer ou la remplacer.

Les leons que l'on peut tirer de cet encodage sont doubles. Tout d'abord, en ce qui concerne l'expressivit de HOcore, il montre comment on peut s'appuyer sur l'ordre suprieur pour atteindre un calcul Turing complet. Les travaux prcdents [5] tudiaient l'expressivit en se basant sur des machines de Minsky, qui n'ont besoin que de savoir compter et de dtecter qu'un compteur vaut 0. L'ordre suprieur n'est pas ncessaire pour compter, il est en revanche utilis pour dtecter l'galit 0 (voir [3] et [1] pour des versions de calculs sans ordre suprieur utilisant d'autres fonctionnalits pour dtecter le 0). La seconde leon porte sur la dcidabilit de la congruence barbue. En effet, nous avons montr prcdemment que la congruence barbue est dcidable pour HOcore [5, 2] si aucun nom de canal n'est cach, et cette congruence devient indcidable si quatre noms de canaux sont cachs. Notre traduction n'ayant besoin que de deux noms de canaux, il permet d'affiner le nombre minimal de restrictions globales ncessaire pour rendre la congruence barbue forte indcidable : il en suffit de deux.

1. Ce travail a bnfici du soutien de l'Agence Nationale pour la Recherche dans le cadre du projet PiCoq ANR 10 BLAN 0305.

Le reste de ce papier est organis comme suit. Nous prsentons le calcul HOcore en section 2, et la KAM en section 3. La traduction et sa preuve de correspondance oprationnelle est prsente en section 4 avant de conclure en section 5.

2. Prsentation de HOcore

2.1. Syntaxe

Le calcul HOcore [5] peut tre vu comme une restriction du π-calcul d'ordre suprieur [7], auquel on aurait enlev l'oprateur de restriction de nom. Il peut galement tre vu comme un λ-calcul parallle, o l'application d'une fonction $\lambda x.Q$ un argument P est remplace par une communication sur un canal a entre un envoi de message $\overline{a}\langle P\rangle$ mis en parallle d'une rception de message $a(x).Q$.

La syntaxe de HOcore est la suivante.

$$P ::= a(x).P \;\Big|\; \overline{a}\langle P\rangle \;\Big|\; P \parallel P \;\Big|\; x \;\Big|\; \mathbf{0}$$

Un processus P peut soit tre une rception de message sur un certain canal, note $a(x).P$, soit une mission de message, note $\overline{a}\langle P\rangle$, soit la mise en parallle de processus $P \parallel Q$, soit une variable x, soit le processus inactif $\mathbf{0}$. Nous distinguons les *variables* x, y, z des *noms de canaux* a, b, c.

Intuitivement, l'unique rgle de rduction est la rgle de communication suivante, o l'opration $[P/x]Q$ est la *substitution* de la variable x par le processus P dans Q. Le processus P, mis sur le canal a, est transmis au prfixe de rception $a(x).Q$.

$$\overline{a}\langle P\rangle \parallel a(x).Q \longrightarrow [P\,/\,x]Q \qquad (\dagger)$$

Notons que le calcul est asynchrone : l'mission de message n'a pas de continuation. Dans un calcul synchrone, l'envoi de message est de la forme $\overline{a}\langle P\rangle.Q$, o le processus Q est la continuation dmarre aprs l'mission du message sur a. Nous montrerons dans la section 4.3 que les messages synchrones permettent un encodage de la KAM ne ncessitant qu'un seul nom de canal.

Variables Une variable x est dite *lie* si elle est sous la porte d'un lieur pour cette variable (une rception de message $a(x).P$), *libre* sinon. Par exemple, dans le processus $a(x).(P \parallel y)$ avec $x \neq y$, les occurrences de x dans P sont lies, mais y est libre.

La machine de Krivine pour le λ-calcul en appel par nom [4] permet de ne considrer que des termes clos, et d'viter les difficults usuelles dues la capture de variables libres. Il en est de mme avec l'encodage que nous prsentons ici : nous ne manipulerons que des termes dont toutes les variables seront lies. Les noms de canaux sont eux tous libres, HOcore n'ayant pas d'oprateur de restriction.

2.2. Smantique

La smantique de HOcore est dfinie par un systme de transitions tiquetes (LTS). Les tiquettes sont dfinies par la syntaxe suivante :

$$\alpha ::= \overline{a}\langle P\rangle \mid a(P) \mid \tau \ .$$

Elles correspondent respectivement l'mission d'un processus, la rception d'un processus, et une communication interne. Le LTS est dfini inductivement, par les rgles suivantes.

$$\frac{}{\overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}}\ \textsc{Out} \qquad \frac{}{a(x).Q \xrightarrow{a(P)} [P\,/\,x]Q}\ \textsc{Inp}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q}\ \textsc{Par1} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'}\ \textsc{Par2}$$

$$\frac{P \xrightarrow{\overline{a}\langle R\rangle} P' \qquad Q \xrightarrow{a(R)} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}\ \textsc{Tau1} \qquad \frac{P \xrightarrow{a(R)} P' \qquad Q \xrightarrow{\overline{a}\langle R\rangle} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}\ \textsc{Tau2}$$

Congruence structurelle Notons que le LTS prcdent est trs rigide : il conserve strictement la structure des termes (l'inverse, une smantique rductionelle utilise gnralement une notion de congruence structurelle, qui permet par exemple de rorganiser les parenthses modulo associativit de la mise en parallle). L'avantage est que les termes obtenus aprs une rduction sont plus facile analyser ; l'inconvnient est que ce LTS tend gnrer de nombreuses occurrences du processus $\mathbf{0}$, qui sont inutiles. Par exemple, on a formellement les transitions suivantes :

$$(\overline{a}\langle P\rangle \parallel \overline{b}\langle Q\rangle) \parallel a(x).b(y).(x \parallel y) \xrightarrow{\tau} (\mathbf{0} \parallel \overline{b}\langle Q\rangle) \parallel b(y).(P \parallel y) \xrightarrow{\tau} (\mathbf{0} \parallel \mathbf{0}) \parallel (P \parallel Q)$$

On va donc s'appuyer sur une notion de congruence structurelle trs restreinte qui permet de s'affranchir de ces occurrences de $\mathbf{0}$.

Une relation $\mathcal{R}$ est une *congruence* si c'est une relation d'quivalence (rflexive, symtrique, transitive) qui respecte les diffrents constructeurs du langage (par exemple si $P\ \mathcal{R}\ Q$ alors nous avons $a(x).P\ \mathcal{R}\ a(x).Q$). On quotiente dans la suite l'ensemble des processus par la plus petite congruence telle que la composition parallle admette le processus $\mathbf{0}$ comme lment neutre gauche et droite.

2.3. Expressivit

HOcore est qualifi de minimal, car il ne contient que le strict ncessaire l'ordre suprieur. Par exemple, il n'inclut pas d'oprateurs de restriction ou de rplication. HOcore est tout de mme Turing complet : un encodage fidle des machines de Minsky est prsent dans [5]. En particulier, le problme de la terminaison y est indcidable.

Un objectif de ce papier est de montrer qu'il est possible d'encoder directement des modles de calcul plus complexes, comme le λ-calcul, dans HOcore.

2.4. quivalence de processus

Une des questions cruciales de l'tude de calculs de processus est de savoir si deux processus « font la mme chose ». Ainsi, dans l'optique de la programmation modulaire, on doit tre capable de dire si deux bibliothques logicielles sont interchangeables. Deux processus sont quivalents si, dans n'importe quel contexte, ce que l'on observe de leur activit est similaire. Formellement, on dfinit la *congruence barbue* [6] comme la plus grande relation symtrique telle que :

- si $P \simeq Q$ et $P \xrightarrow{\tau} P'$, alors il existe un Q' tel que $Q \xrightarrow{\tau} Q'$ et $P' \simeq Q'$: la congruence barbue est prserve par rductions ;
- si $P \simeq Q$, alors $C[P] \simeq C[Q]$ pour tout contexte C, un contexte tant un processus avec un trou : la congruence barbue est une congruence ;

– si $P \simeq Q$ et $P \xrightarrow{\overline{a}\langle P''\rangle} P'$, alors il existe Q' et Q'' tels que $Q \xrightarrow{\overline{a}\langle Q''\rangle} Q'$: la congruence barbue met en relation des processus avec les mmes *observables*, ou barbes, qui sont ici la possibilit d'mettre un message sur un canal donn.

On peut dfinir de faon similaire la *congruence barbue faible*, note $\cong$, en considrant dans la premiere clause des squences arbitraires de transitions, plutt que des transitions simples.

Un des rsultats fondamentaux de HOcore est la dcidabilit de la congruence barbue [5]. Les processus de HOcore ne pouvant pas cacher leurs rductions (le langage n'a pas d'oprateur de restriction), il est toujours possible de dfinir des contextes explorant la structure d'un processus. En revanche, ds que l'on autorise des rductions anonymes, par exemple grce des restrictions globales empchant l'observation sur certains noms, la congruence barbue devient indcidable. Les travaux prcdents ont montr que quatre telles restrictions globales suffisent.

3. La KAM

La machine abstraite de Krivine est une machine trs simple pour valuer les termes du λ-calcul en appel par nom. Elle permet galement de dfinir des oprateurs de contrle.

Une configuration de la KAM est compos d'un terme du λ-calcul et d'une pile, qui est une liste de λ-termes. Comme indiqu plus haut, tous les λ-termes considrs sont *clos* (ils ne contiennent pas de variable libre).

$$C ::= M \star \pi$$
$$M ::= x \mid MN \mid \lambda x.M$$
$$\pi ::= M :: \pi \mid []$$

Les rgles de rductions de la KAM (sans oprateur de contrle) sont les suivantes ; un calcul s'arrte quand un tat $\lambda x.M \star []$ est atteint.

$$MN \star \pi \mapsto M \star N :: \pi \qquad \text{(Push)}$$
$$\lambda x.M \star N :: \pi \mapsto [N\,/\,x]M \star \pi \qquad \text{(Grab)}$$

La KAM avec oprateurs de contrles ajoute deux constructions syntaxiques : l'oprateur de capture de continuation `cc`, utilisable dans les programmes, et les constantes de pile k_π, qui ne peuvent tre cres que par appel `cc`. Les deux rgles suivantes sont alors ajoutes.

$$\texttt{cc} \star M :: \pi \mapsto M \star k_\pi :: \pi \qquad \text{(CallCC)}$$
$$k_\pi \star M :: \pi' \mapsto M \star \pi \qquad \text{(Restore)}$$

4. KAM en HOcore

Nous commenons par traduire la KAM sans oprateurs de contrle ; nous tendons ensuite notre encodage ces oprateurs en section 4.2.

4.1. Version asynchrone

Nous prsentons un codage n'utilisant que deux noms libres. La pile courante est un message sur le nom c (“c” pour *continuation*). Le contenu de la pile est un message sur le nom a (“a” comme

argument), correspondant la tte de la pile, et un message sur c contenant la queue de la pile. La traduction de la pile vide est arbitrairement fixe au processus $\overline{b}\langle\mathbf{0}\rangle$, pour un troisime nom libre b permettant d'observer la fin du calcul (les rsultats dmontrs ci-dessous sont toujours valides lorsque ce processus est remplac par un processus arbitraire, tant que celui-ci n'introduit pas de divergence ou de non-dterminisme).

La traduction d'une pile $1 :: 2 :: 3 :: []$ prend donc la forme $\overline{a}\langle 1\rangle \parallel \overline{c}\langle\overline{a}\langle 2\rangle \parallel \overline{c}\langle\overline{a}\langle 3\rangle \parallel \overline{c}\langle\overline{b}\langle\mathbf{0}\rangle\rangle\rangle\rangle$.

Pour ne pas alourdir les notations, nous utilisons le mme symbole pour traduire des tats, des piles et des λ-termes. Nous supposons galement que les noms des variables lies u et s (u n'apparat que pour la traduction des oprateurs de contrle) sont diffrents des noms de variables du λ-terme.

$$
\begin{aligned}
[\![[]]\!] &\triangleq \overline{b}\langle\mathbf{0}\rangle \\
[\![M :: \pi]\!] &\triangleq \overline{a}\langle[\![M]\!]\rangle \parallel \overline{c}\langle[\![\pi]\!]\rangle \\
[\![MN]\!] &\triangleq c(s).([\![M]\!] \parallel \overline{c}\langle\overline{a}\langle[\![N]\!]\rangle \parallel \overline{c}\langle s\rangle\rangle) \\
[\![\lambda x.M]\!] &\triangleq c(s).(a(x).\,[\![M]\!] \parallel s) \\
[\![x]\!] &\triangleq x \\
[\![M \star \pi]\!] &\triangleq [\![M]\!] \parallel \overline{c}\langle[\![\pi]\!]\rangle
\end{aligned}
$$

Notons que dans la traduction d'une configuration, la pile, compose de messages imbriqus sur a et c, est elle mme encapsule l'intrieur d'un message sur le nom c.

Nous montrons ci-dessous que la rduction de $[\![M \star \pi]\!]$ est dterministe pour tout M et tout π. De plus, chaque tape de calcul de la KAM correspond une ou deux tapes de calcul du processus traduit (ou trois lorsque l'on prend en compte les oprateurs de contrle).

Lemme 1 (Substitution)**.** *Pour tous M, x, N avec N clos, nous avons $[\![[N\,/\,x]M]\!] = [[\![N]\!]\,/\,x]\,[\![M]\!]$.*

Démonstration. Par une simple induction sur M. □

Lemme 2 (Simulation)**.** *Pour tous M, M', π, π' tels que $M \star \pi \mapsto M' \star \pi'$, nous avons $[\![M \star \pi]\!] \xrightarrow{\tau}{}^{+} [\![M' \star \pi']\!]$.*

Démonstration. Il suffit de considrer les rgles de rduction de la machine de Krivine :

PUSH ($MN \star \pi \mapsto M \star N :: \pi$) : on vrifie que

$$
\begin{aligned}
[\![MN \star \pi]\!] &= (c(s).\,[\![M]\!] \parallel \overline{c}\langle\overline{a}\langle[\![N]\!]\rangle \parallel \overline{c}\langle s\rangle\rangle) \parallel \overline{c}\langle[\![\pi]\!]\rangle \\
&\xrightarrow{\tau} [\![M]\!] \parallel \overline{c}\langle\overline{a}\langle[\![N]\!]\rangle \parallel \overline{c}\langle[\![\pi]\!]\rangle\rangle \\
&= [\![M \star N :: \pi]\!] \quad .
\end{aligned}
$$

Il suffit donc d'une seule transition pour simuler cette rgle.

GRAB ($\lambda x.M \star N :: \pi \mapsto [N\,/\,x]M \star \pi$) : il faut cette fois deux transitions :

$$
\begin{aligned}
[\![\lambda x.M \star N :: \pi]\!] &= (c(s).(a(x).\,[\![M]\!]) \parallel s) \parallel \overline{c}\langle\overline{a}\langle[\![N]\!]\rangle \parallel \overline{c}\langle[\![\pi]\!]\rangle\rangle \\
&\xrightarrow{\tau} (a(x).\,[\![M]\!]) \parallel \overline{a}\langle[\![N]\!]\rangle \parallel \overline{c}\langle[\![\pi]\!]\rangle \\
&\xrightarrow{\tau} [[\![N]\!]\,/\,x]\,[\![M]\!] \parallel \overline{c}\langle[\![\pi]\!]\rangle \\
&= [\![[N\,/\,x]M]\!] \parallel \overline{c}\langle[\![\pi]\!]\rangle \qquad \text{(par le Lemme 1)} \\
&= [\![[N\,/\,x]M \star \pi]\!] \quad .
\end{aligned}
$$

□

Il nous faut maintenant prouver que les transitions des processus traduits sont dterministes, et qu'elles correspondent des rductions dans la KAM. Or, comme on le voit dans la preuve prcdente, certaines transitions sont intermdiaires et doivent tre compltes afin de correspondre prcisment une rduction de la KAM.

Afin de simplifier la preuve, nous passons par une machine abstraite lgrement diffrente de la KAM, dans laquelle certaines tapes sont artificiellement ddoubles : les tapes de calcul des processus traduits correspondent ainsi exactement aux rductions de la KAM modifie.

La modification est la suivante : on introduit une configuration intermdiaire, note $\lambda' x.M \star \pi$, et la rgle GRAB est ddouble comme suit :

$$\lambda x.M \star \pi \mapsto \lambda' x.M \star \pi \qquad (\text{GRAB}_1)$$

$$\lambda' x.M \star N :: \pi \mapsto [N \,/\, x]M \star \pi \qquad (\text{GRAB}_2)$$

Fait 3. *Une configuration admet une squence infinie de rductions dans la KAM originelle si et seulement elle admet une squence infinie de rductions dans la KAM modifie.*

En accord avec le second cas dans la preuve du lemme 2, la fonction de traduction est tendue aux configurations intermdiaires en posant :

$$\llbracket \lambda' x.M \star \pi \rrbracket \stackrel{\triangle}{=} (a(x).\llbracket M \rrbracket) \parallel \llbracket \pi \rrbracket \quad .$$

Lemme 4. *La fonction de traduction est injective.*

Démonstration. On prouve qu'elle est injective sur les λ-termes, puis sur les piles, puis sur les configurations. □

Notons $\llbracket \cdot \rrbracket^{-1}$ la fonction partielle inverse de la traduction, i.e., telle que $\llbracket \llbracket C \rrbracket \rrbracket^{-1} = C$ pour toute configuration C de la KAM modifie. On se convainc aisment que cette fonction est calculable.

Lemme 5 (Rflection)**.** *Pour toute configuration C et processus P, si $\llbracket C \rrbracket \xrightarrow{\tau} P$, alors $\llbracket P \rrbracket^{-1}$ est dfini et $C \mapsto \llbracket P \rrbracket^{-1}$.*

Démonstration. On raisonne par cas sur la configuration C :

- $C = MN \star \pi$: on a $\llbracket C \rrbracket = (c(s).\llbracket M \rrbracket \parallel \overline{c}\langle \overline{a}\langle \llbracket N \rrbracket \rangle \parallel \overline{c}\langle s \rangle \rangle) \parallel \overline{c}\langle \llbracket \pi \rrbracket \rangle$, d'o $P = \llbracket M \rrbracket \parallel \overline{c}\langle \overline{a}\langle \llbracket N \rrbracket \rangle \parallel \overline{c}\langle \llbracket \pi \rrbracket \rangle \rangle$ puisqu'une seule transition est possible. On vrifie alors que $\llbracket P \rrbracket^{-1} = M \star N :: \pi$, et $C \mapsto \llbracket P \rrbracket^{-1}$ par la rgle (PUSH).
- $C = \lambda x.MN \star \pi$: on a $\llbracket C \rrbracket = (c(s).(a(x).\llbracket M \rrbracket) \parallel s) \parallel \overline{c}\langle \llbracket \pi \rrbracket \rangle$, d'o $P = (a(x).\llbracket M \rrbracket) \parallel \llbracket \pi \rrbracket$. On vrifie alors que $\llbracket P \rrbracket^{-1} = \lambda' x.M \star \pi$, et $C \mapsto \llbracket P \rrbracket^{-1}$ par la rgle (GRAB_1).
- $C = \lambda' x.MN \star \pi$: si la pile π est vide, alors $\llbracket C \rrbracket = (a(x).\llbracket M \rrbracket) \parallel \overline{b}\langle \mathbf{0} \rangle$ n'admet pas de transition τ, ce qui contredit l'hypothse sur P. On a donc $\pi = N :: \pi'$, et $\llbracket C \rrbracket = (a(x).\llbracket M \rrbracket) \parallel \overline{a}\langle N \rangle \parallel \overline{c}\langle \llbracket \pi' \rrbracket \rangle$. On a ncessairement $P = [\llbracket N \rrbracket / x]\,\llbracket M \rrbracket \parallel \overline{c}\langle \llbracket \pi' \rrbracket \rangle$; on vrifie alors que $\llbracket P \rrbracket^{-1} = [N/x]M \star \pi'$ l'aide du Lemme 1, et que $C \mapsto \llbracket P \rrbracket^{-1}$ par la rgle (GRAB_2). □

Thorme 6 (Dterminisme)**.** *Pour toute configuration C et tous processus P, P', P'', si $\llbracket C \rrbracket \xrightarrow{\tau}{}^{\star} P$, $P \xrightarrow{\tau} P'$ et $P \xrightarrow{\tau} P''$, alors $P' = P''$.*

Démonstration. Par le Lemme 5, on peut se ramener au cas o $P = \llbracket C \rrbracket$. En reprenant la preuve de ce mme lemme, par analyse de cas sur C, on constate qu'au plus une communication est possible dans le processus traduit $\llbracket C \rrbracket$. □

Thorme 7 (Correspondance oprationelle)**.** *Pour toutes configurations C, C', $C \mapsto C'$ si et seulement si $[\![C]\!] \xrightarrow{\tau} [\![C']\!]$.*

Démonstration. Consquence immdiate des Lemmes 2 et 5. □

Thorme 8. *Pour toute configuration C, nous avons C termine si et seulement si $[\![C]\!] \xrightarrow{\tau}{}^{\star} \xrightarrow{\overline{b}\langle \mathbf{0} \rangle}$.*

Démonstration. La machine abstraite s'arrte exactement sur les configurations de la forme $\lambda' x.M \star []$, dont les encodages sont de la forme $(a(x).[\![M]\!]) \parallel \overline{b}\langle \mathbf{0} \rangle$, qui ont pas de transitions τ, et qui ont une barbe sur b. Inversement, les seules configurations dont la traduction est capable d'mettre sur b sont celles de la forme $\lambda' x.M \star []$. On peut donc conclure par le Thorme 7. □

En tant que fragment du π-calcul d'ordre suprieur [7], HOcore peut naturellement tre tendu en ajoutant un oprateur de restriction de nom. Nous ne considrons ici qu'une extension plus rduite n'utilisant que des restrictions globales (i.e., les restrictions ne peuvent tre dans des messages, donc elles ne peuvent tre rpliques). La syntaxe est tendue de la manire suivante.

$$T ::= \nu a.T \mid P$$

L'extension du LTS est immdiate : les seules transitions autorises pour les termes $\nu a.T$ sont celles de T dont les noms ne mentionnent pas a. Nous notons $\mathtt{n}(\alpha)$ les noms de canaux de α.

$$\frac{T \xrightarrow{\alpha} T' \qquad a \notin \mathtt{n}(\alpha)}{\nu a.T \xrightarrow{\alpha} \nu a.T'}$$

Intuitivement, $\nu a.P$ correspond au processus P auquel on interdit de communiquer sur a avec l'extrieur (en mission comme en rception) : le nom de canal a est connu de lui seul, toutes les communications sur a ne peuvent donc avoir lieu qu' l'intrieur de P.

Thorme 9. *Soit $\Omega \triangleq \nu a.\overline{a}\langle a(x).(\overline{a}\langle x \rangle \parallel x) \rangle \parallel a(x).(\overline{a}\langle x \rangle \parallel x)$. Pour tout λ-terme M et toute pile π, on a $\nu a.\nu c.[\![M \star \pi]\!] \simeq \Omega$ si et seulement si $M \star \pi$ ne termine pas.*

Démonstration. Ω est un processus divergent, dont la seule transition est $\Omega \xrightarrow{\tau} \Omega$. Par consquent, si $M \star \pi$ ne termine pas, alors $\nu a.\nu c.[\![M \star \pi]\!]$ lui est quivalent, puisqu'il ne pourra jamais mettre sur son seul nom libre (b). Inversement, si $M \star \pi$ termine, alors $\nu a.\nu c.[\![M \star \pi]\!]$ finira par mettre sur b, ce qui le distingue du processus Ω. □

Corollaire 10. *L'quivalence contextuelle est indcidable dans HOcore avec deux restrictions globales.*

Un raisonnement similaire permet d'obtenir l'indcidabilit de l'quivalence contextuelle faible dans HOcore avec deux restrictions globales ; on peut mme utiliser dans ce cas le processus vide, $\mathbf{0}$, plutt que le processus divergent Ω (notons cependant que pour cette preuve, on n'a pas la possibilit d'encoder la pile vide par un processus arbitraire ; en particulier, le processus vide ne conviendrait pas : il est ncessaire d'avoir une observable visible lorsque le fond de pile est atteint). En contrepartie, dans le cas fort, le fond de pile peut tre traduit par $\mathbf{0}$ en effectuant la comparaison avec Ω.

4.2. Oprateurs de contrle

Nous ajoutons maintenant les oprateurs de contrle (call-cc). Rappelons les deux rgles de rduction de la KAM dfinissant ces oprateurs :

$$\mathtt{cc} \star M :: \pi \mapsto M \star k_\pi :: \pi \qquad \text{(CallCC)}$$
$$k_\pi \star M :: \pi' \mapsto M \star \pi \qquad \text{(Restore)}$$

Etant donn un processus P, on dfinit

$$K(P) \triangleq c(s_0).(s_0 \parallel a(u).c(\_).(u \parallel \overline{c}\langle P\rangle)).$$

Les deux oprateurs sont alors traduits comme suit :

$$\begin{aligned}[\![\mathtt{cc}]\!] &\triangleq c(s_0).(s_0 \parallel c(s).a(u).(u \parallel \overline{c}\langle\overline{a}\langle K(s)\rangle \parallel \overline{c}\langle s\rangle\rangle))\\ [\![k_\pi]\!] &\triangleq K([\![\pi]\!])\end{aligned}$$

Cet encodage ncessite trois transitions pour simuler les rgles (CallCC) et (Restore) de la KAM avec oprateurs de contrle. Comme prcdemment pour la rgle (Grab), on introduit donc quatre configurations intermdiaires et artificielles dans la KAM, dont la smantique est dfinie par les six rgles suivantes.

$$\mathtt{cc} \star \pi \mapsto \mathtt{cc}_1 \star \pi \qquad (\text{CallCC}_1)$$
$$\mathtt{cc}_1 \star M :: \pi \mapsto \mathtt{cc}_2 \star M :: \pi \qquad (\text{CallCC}_2)$$
$$\mathtt{cc}_2 \star M :: \pi \mapsto M \star k_\pi :: \pi \qquad (\text{CallCC}_3)$$

$$k_\pi \star \pi' \mapsto k_\pi^1 \star \pi' \qquad (\text{Restore}_1)$$
$$k_\pi^1 \star M :: \pi' \mapsto k_\pi^2 \star M :: \pi' \qquad (\text{Restore}_2)$$
$$k_\pi^2 \star M :: \pi' \mapsto M \star \pi \qquad (\text{Restore}_3)$$

Ces quatre configurations s'encodent comme suit dans HOcore.

$$\begin{aligned}[\![\mathtt{cc}_1 \star \pi]\!] &\triangleq [\![\pi]\!] \parallel c(s).a(u).(u \parallel \overline{c}\langle\overline{a}\langle K(s)\rangle \parallel \overline{c}\langle s\rangle\rangle)\\ [\![\mathtt{cc}_2 \star M :: \pi]\!] &\triangleq \overline{a}\langle[\![M]\!]\rangle \parallel a(u).(u \parallel \overline{c}\langle\overline{a}\langle K([\![\pi]\!])\rangle \parallel \overline{c}\langle[\![\pi]\!]\rangle\rangle)\\ &= \overline{a}\langle[\![M]\!]\rangle \parallel a(u).(u \parallel \overline{c}\langle\overline{a}\langle[\![k_\pi]\!]\rangle \parallel \overline{c}\langle[\![\pi]\!]\rangle\rangle)\\ &= \overline{a}\langle[\![M]\!]\rangle \parallel a(u).(u \parallel \overline{c}\langle[\![k_\pi :: \pi]\!]\rangle)\\ [\![k_\pi^1 \star \pi']\!] &\triangleq [\![\pi']\!] \parallel a(u).c(\_).(u \parallel \overline{c}\langle[\![\pi]\!]\rangle)\\ [\![k_\pi^2 \star M :: \pi']\!] &\triangleq \overline{c}\langle[\![\pi']\!]\rangle \parallel c(\_).([\![M]\!] \parallel \overline{c}\langle[\![\pi]\!]\rangle)\\ &= \overline{c}\langle[\![\pi']\!]\rangle \parallel c(\_).\,[\![M \star \pi]\!]\end{aligned}$$

On vrifie alors que les processus traduits correspondants ont exactement les transitions suivantes.

$$[\![\mathtt{cc} \star M :: \pi]\!] \xrightarrow{\tau} [\![\mathtt{cc}_1 \star M :: \pi]\!] \xrightarrow{\tau} [\![\mathtt{cc}_2 \star M :: \pi]\!] \xrightarrow{\tau} [\![M \star k_\pi :: \pi]\!]$$
$$[\![k_\pi \star M :: \pi']\!] \xrightarrow{\tau} [\![k_\pi^1 \star M :: \pi']\!] \xrightarrow{\tau} [\![k_\pi^2 \star M :: \pi']\!] \xrightarrow{\tau} [\![M \star \pi]\!]$$

Les preuves des lemmes 1, 2, 4 et 5 ainsi que des thormes 6 et 7 s'tendent sans difficult.

Le lecteur attentif aura remarqu que la traduction de l'oprateur cc effectue d'abord une rception $c(s)$ avant la rception $a(u)$. Ce choix est purement esthtique : changer l'ordre des rceptions est tout fait possible, mais ne permet pas d'utiliser $[\![k_\pi :: \pi]\!]$ dans la traduction de $\mathtt{cc}_2$, la rendant moins succincte.

4.3. Version synchrone

Nous montrons maintenant qu'il est possible d'obtenir une traduction de la KAM en utilisant un seul nom, si le calcul considr est *synchrone*. Une version synchrone de HOcore ne modifie que l'mission de message, remplaant la construction $\overline{a}\langle P\rangle$ par $\overline{a}\langle P\rangle.Q$. Cette dernire met toujours un message sur a

transportant P. En revanche, ds que le message est reu, le processus Q, appel *continuation*, est lanc : la rgle de rduction intuitive devient

$$\overline{a}\langle P\rangle.Q \parallel a(x).R \longrightarrow Q \parallel [P\,/\,x]R \qquad (\ddagger)$$

Formellement, cela se traduit dans le LTS par une simple modification de la rgle axiome Out :

$$\frac{}{\overline{a}\langle P\rangle.Q \xrightarrow{\overline{a}\langle P\rangle} Q}\ \textsc{Out}$$

La traduction de la KAM (tendue) est donne ci-dessous. Nous traduisons dsormais les piles comme tant des messages synchrones : le contenu du message tant la tte de la pile et la continuation du message la queue de la pile. Nous traduisons ainsi une pile $1 :: 2 :: 3 :: []$ par $\overline{a}\langle 1\rangle.\overline{a}\langle\overline{a}\langle 2\rangle.\overline{a}\langle\overline{a}\langle 3\rangle.\overline{a}\langle\overline{b}\langle\mathbf{0}\rangle\rangle\rangle\rangle$.

Au sein d'une configuration, comme dans le cas asynchrone, la pile sera de plus encapsule l'intrieur d'un message additionel sur a.

Comme ci-dessus, pour tout processus P, nous dfinissons

$$K(P) \triangleq a(s_0).(s_0 \parallel a(u).a(\_).(u \parallel \overline{a}\langle P\rangle)).$$

(Par convention, nous notons $\overline{a}\langle P\rangle$ les messages $\overline{a}\langle P\rangle.\mathbf{0}$ dont la continuation est vide). Nous dfinissons alors la traduction synchrone comme suit.

$$\begin{aligned}
[\![[]]\!] &\triangleq \overline{b}\langle\mathbf{0}\rangle\\
[\![M :: \pi]\!] &\triangleq \overline{a}\langle[\![M]\!]\rangle.\overline{a}\langle[\![\pi]\!]\rangle\\
[\![MN]\!] &\triangleq a(s).([\![M]\!] \parallel \overline{a}\langle\overline{a}\langle[\![N]\!]\rangle.\overline{a}\langle s\rangle\rangle)\\
[\![\lambda x.M]\!] &\triangleq a(s).(a(x).\,[\![M]\!] \parallel s)\\
[\![x]\!] &\triangleq x\\
[\![M \star \pi]\!] &\triangleq [\![M]\!] \parallel \overline{a}\langle[\![\pi]\!]\rangle\\
[\![\mathtt{cc}]\!] &\triangleq a(s_0).(s_0 \parallel a(u).a(s).(u \parallel \overline{a}\langle\overline{a}\langle K(s)\rangle.\overline{a}\langle s\rangle\rangle))\\
[\![k_\pi]\!] &\triangleq K([\![\pi]\!])
\end{aligned}$$

A nouveau, les preuves de la section 4.1 s'adaptent sans difficult. L'quivalence contextuelle est donc indcidable dans HOcore synchrone avec une seule restriction de nom globale.

5. Conclusion

Nous avons prsent deux traductions directes de la machine abstraite de Krivine avec oprateurs de contrle dans HOcore, utilisant un nom libre dans le cas synchrone, et deux dans le cas asynchrone. Cela nous a permis d'affiner la borne sur le nombre de restrictions globales de noms de canaux suffisant pour obtenir l'indcidabilit des quivalences contextuelles fortes et faible (en l'absence de restriction de nom, l'quivalence contextuelle forte est dcidable que le calcul soit synchrone ou asynchrone [5]—le cas faible est ouvert).

Nous pouvons remarquer que les trois fonctionnalits fondamentales utilises pour traduire la KAM sont les suivantes. Les deux premires portent sur l'ordre suprieur : les messages transportent des processus, et la rception effectue une substitution identique celle du λ-calcul. La troisime fonctionnalit sur laquelle nous nous appuyons porte sur le contrle de l'valuation : nous devons distinguer entre la tte de la pile et le reste de la pile. Pour ce faire, nous pouvons utiliser deux noms diffrents (version asynchrone), ou squentialiser les missions (version synchrone) et n'utiliser ainsi qu'un seul nom. Nous

ne pensons pas qu'il soit possible d'obtenir une traduction de la KAM avec un seul nom dans un calcul asynchrone.

La relative simplicit de notre traduction, et le fait qu'elle permette de traiter immdiatement les oprateurs de contrle, nous laisse esprer qu'elle permettra une meilleure comprhension de l'expressivit du calcul HOcore, et peut-tre, terme, de rsoudre la question de la dcidabilit de l'quivalence contextuelle faible—dans le calcul sans restriction de nom.

Références

[1] J. Aranda, F. D. Valencia, and C. Versari. On the expressive power of restriction and priorities in ccs with replication. In L. Alfaro, editor, *Foundations of Software Science and Computational Structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg, 2009.

[2] S. Boulier and A. Schmitt. Formalisation de hocore en coq. In *Actes des 23èmes Journées Francophones des Langages Applicatifs*, Jan. 2012.

[3] N. Busi, M. Gabbrielli, and G. Zavattaro. On the expressive power of recursion, replication and iteration in process calculi. *Mathematical Structures in Computer Science*, 19(6) :1191–1222, Dec. 2009.

[4] J.-L. Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3) :199–207, 2007.

[5] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. *Information and Computation*, 209(2) :198–226, Feb. 2011. Extended abstract presented at *Logic in Computer Science (LICS)*, 2008.

[6] R. Milner and D. Sangiorgi. Barbed bisimulation. In *19th ICALP*, volume 623 of *LNCS*, pages 685–695. Springer Verlag, 1992.

[7] D. Sangiorgi and D. Walker. *The π-calculus : a Theory of Mobile Processes*. Cambridge University Press, 2001.