



De la KAM avec un Processus d'Ordre Supérieur

Damien Pous, Alan Schmitt

► **To cite this version:**

Damien Pous, Alan Schmitt. De la KAM avec un Processus d'Ordre Supérieur. JFLAs 2014, Jan 2014, Fréjus, France. pp.1-12, 2014. <hal-00966097v1>

HAL Id: hal-00966097

<https://hal.archives-ouvertes.fr/hal-00966097v1>

Submitted on 26 Mar 2014 (v1), last revised 27 Jan 2016 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

De la KAM avec un Processus d'Ordre Supérieur

D. Pous¹ & A. Schmitt²

1: CNRS, damien.pous@ens-lyon.fr

2: Inria, alan.schmitt@inria.fr

Résumé

Nous présentons un encodage simple et direct de la machine abstraite de Krivine (KAM) dans le calcul de processus d'ordre supérieur HOcore, en utilisant un nombre très restreint de canaux de communication. Cet encodage montre qu'il est possible de capturer l'expressivité du λ -calcul en HOcore dès que l'on fixe l'ordre d'évaluation. Nous donnons également une nouvelle borne inférieure pour le nombre minimal de restrictions nécessaire pour rendre l'équivalence de programmes dans HOcore décidable.¹

1. Introduction

Le calcul de processus HOcore est remarquable par sa similarité au λ -calcul, auquel il ajoute une notion de concurrence. Malgré cette similarité, aucun encodage du λ -calcul en HOcore n'a été présenté jusqu'à présent. En effet, l'appariement entre une fonction et son argument est très syntaxique et très rigide en λ -calcul, alors qu'il est beaucoup plus lâche en HOcore car il correspond à la présence simultanée sur le même nom de canal d'un message et d'un récepteur pour ce message. Cette différence cruciale, couple l'impossibilité de générer de nouveaux noms de canaux, implique que toute traduction doit fixer le nombre de redex pouvant être actifs en parallèle. Ce nombre pouvant être non-borné pour certains λ -termes, ceci empêche toute traduction *tant que la stratégie d'évaluation n'a pas été fixée*. C'est cette deuxième voie que nous explorons ici, en choisissant une stratégie en appel par nom, décrite sous la forme d'une machine abstraite de Krivine (KAM).

Une fois ce choix de conception arrêté, la traduction est très naturelle : on représente la structure récursive de la pile de la KAM comme deux messages transportant respectivement le terme de tête et, récursivement, la queue de la pile. La β -réduction est simplement la communication entre le terme actif, représentant la fonction, avec la tête de la pile. De manière surprenante, cette traduction permet également de capturer l'opérateur de contrôle call-cc de la KAM. La réduction de la continuation en tant que pile contenue dans un message permet en effet de facilement la dupliquer ou la remplacer.

Les leçons que l'on peut tirer de cet encodage sont doubles. Tout d'abord, en ce qui concerne l'expressivité de HOcore, il montre comment on peut s'appuyer sur l'ordre supérieur pour atteindre un calcul Turing complet. Les travaux précédents [5] jugeaient l'expressivité en se basant sur des machines de Minsky, qui n'ont besoin que de savoir compter et de détecter qu'un compteur vaut 0. L'ordre supérieur n'est pas nécessaire pour compter, il est en revanche utilisé pour détecter l'égalité 0 (voir [3] et [1] pour des versions de calculs sans ordre supérieur utilisant d'autres fonctionnalités pour détecter le 0). La seconde leçon porte sur la décidabilité de la congruence barbuée. En effet, nous avons montré précédemment que la congruence barbuée est décidable pour HOcore [5, 2] si aucun nom de canal n'est caché, et cette congruence devient indécidable si quatre noms de canaux sont cachés. Notre traduction n'ayant besoin que de deux noms de canaux, il permet d'affiner le nombre minimal de restrictions globales nécessaire pour rendre la congruence barbuée forte décidable : il en suffit de deux.

1. Ce travail a bénéficié 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 presentons le calcul HOcore en section 2, et la KAM en section 3. La traduction et sa preuve de correspondance operationnelle est presente 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 $\bar{a}\langle P \rangle$ mis en parallle d'une rception de message $a(x).Q$.

La syntaxe de HOcore est la suivante.

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

Un processus P peut soit tre une rception de message sur un certain canal, note $a(x).P$, soit une mission de message, note $\bar{a}\langle P \rangle$, 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'operation $[P/x]Q$ 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 $a(x).Q$.

$$\bar{a}\langle P \rangle \parallel a(x).Q \longrightarrow [P/x]Q \quad (\dagger)$$

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 $\bar{a}\langle P \rangle.Q$, 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 presentons 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 transitionstiquetes (LTS). Les tiquettes sont dfinies par la syntaxe suivante :

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

Elles correspondent respectivement l'émission d'un processus, la réception d'un processus, et une communication interne. Le LTS est défini inductivement, par les règles suivantes.

$$\begin{array}{c}
 \frac{}{\bar{a}\langle P \rangle \xrightarrow{} \mathbf{0}} \text{OUT} \qquad \frac{}{a(x).Q \xrightarrow{a(P)} [P/x]Q} \text{INP} \\
 \\
 \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \text{PAR1} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'} \text{PAR2} \\
 \\
 \frac{P \xrightarrow{\bar{a}\langle R \rangle} P' \quad Q \xrightarrow{a(R)} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \text{TAU1} \qquad \frac{P \xrightarrow{a(R)} P' \quad Q \xrightarrow{\bar{a}\langle R \rangle} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \text{TAU2}
 \end{array}$$

Congruence structurelle Notons que le LTS précédent est très rigide : il conserve strictement la structure des termes (l'inverse, une sémantique réductionnelle utilise généralement une notion de congruence structurelle, qui permet par exemple de réorganiser les parenthèses modulo associativité de la mise en parallèle). L'avantage est que les termes obtenus après une réduction sont plus faciles à analyser ; l'inconvénient est que ce LTS tend à générer de nombreuses occurrences du processus $\mathbf{0}$, qui sont inutiles. Par exemple, on a formellement les transitions suivantes :

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

On va donc s'appuyer sur une notion de congruence structurelle très 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'équivalence (réflexive, symétrique, transitive) qui respecte les différents 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 parallèle admette le processus $\mathbf{0}$ comme élément neutre gauche et droite.

2.3. Expressivité

HOcore est qualifié de minimal, car il ne contient que le strict nécessaire l'ordre supérieur. Par exemple, il n'inclut pas d'opérateurs de restriction ou de réplication. HOcore est tout de même Turing complet : un encodage fidèle des machines de Minsky est présent dans [5]. En particulier, le problème de la terminaison y est indécidable.

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

2.4. équivalence de processus

Une des questions cruciales de l'étude de calculs de processus est de savoir si deux processus « font la même chose ». Ainsi, dans l'optique de la programmation modulaire, on doit être capable de dire si deux bibliothèques logicielles sont interchangeables. Deux processus sont équivalents si, dans n'importe quel contexte, ce que l'on observe de leur activité est similaire. Formellement, on définit la *congruence barbue* [6] comme la plus grande relation symétrique 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 préservée par réductions ;
- 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 $P \xrightarrow{\bar{a}\langle P'' \rangle} P'$, alors il existe Q' et Q'' tels que $Q \xrightarrow{\bar{a}\langle Q'' \rangle} Q'$: 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 \approx , 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 opérateurs 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).

$$\begin{aligned} C &::= M \star \pi \\ M &::= x \mid MN \mid \lambda x.M \\ \pi &::= M :: \pi \mid [] \end{aligned}$$

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

$$\begin{aligned} MN \star \pi &\mapsto M \star N :: \pi && \text{(PUSH)} \\ \lambda x.M \star N :: \pi &\mapsto [N / x]M \star \pi && \text{(GRAB)} \end{aligned}$$

La KAM avec opérateurs de contrles ajoute deux constructions syntaxiques : l'opérateur 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 ajoutées.

$$\begin{aligned} cc \star M :: \pi &\mapsto M \star k_\pi :: \pi && \text{(CALLCC)} \\ k_\pi \star M :: \pi' &\mapsto M \star \pi && \text{(RESTORE)} \end{aligned}$$

4. KAM en HOcore

Nous commenons par traduire la KAM sans opérateurs de contrle ; nous tendons ensuite notre encodage ces opérateurs 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 $\bar{b}\langle 0 \rangle$, pour un troisieme 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-determinisme).

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

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 opérateurs de contrle) sont diffrents des noms de variables du λ -terme.

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

Notons que dans la traduction d'une configuration, la pile, compose de messages imbriques 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 $\llbracket M \star \pi \rrbracket$ 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 opérateurs de contrle).

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

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 $\llbracket M \star \pi \rrbracket \xrightarrow{\tau} \llbracket M' \star \pi' \rrbracket$.*

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

$$\begin{aligned} \llbracket MN \star \pi \rrbracket &= (c(s). \llbracket M \rrbracket \parallel \bar{c}\langle \bar{a}\langle \llbracket N \rrbracket \rangle \parallel \bar{c}\langle s \rangle \rangle) \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle \\ &\xrightarrow{\tau} \llbracket M \rrbracket \parallel \bar{c}\langle \bar{a}\langle \llbracket N \rrbracket \rangle \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle \rangle \\ &= \llbracket M \star N :: \pi \rrbracket . \end{aligned}$$

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 :

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

□

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 :

$$\begin{aligned} \lambda x.M \star \pi &\mapsto \lambda'x.M \star \pi && \text{(GRAB}_1\text{)} \\ \lambda'x.M \star N :: \pi &\mapsto [N/x]M \star \pi && \text{(GRAB}_2\text{)} \end{aligned}$$

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 :

$$\llbracket \lambda'x.M \star \pi \rrbracket \triangleq (a(x). \llbracket M \rrbracket) \parallel \llbracket \pi \rrbracket .$$

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 aisement 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 $\llbracket C \rrbracket = (c(s). \llbracket M \rrbracket \parallel \bar{c}\langle \bar{a}\langle [N] \rangle \rangle \parallel \bar{c}\langle s \rangle \rangle) \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle$, d'o $P = \llbracket M \rrbracket \parallel \bar{c}\langle \bar{a}\langle [N] \rangle \rangle \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle$ 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 $\llbracket C \rrbracket = (c(s).(a(x). \llbracket M \rrbracket) \parallel s) \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle$, 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 (GRAB₁).
- $C = \lambda'x.MN \star \pi$: si la pile π est vide, alors $\llbracket C \rrbracket = (a(x). \llbracket M \rrbracket) \parallel \bar{b}\langle \mathbf{0} \rangle$ n'admet pas de transition τ , ce qui contredit l'hypothse sur P . On a donc $\pi = N :: \pi'$, et $\llbracket C \rrbracket = (a(x). \llbracket M \rrbracket) \parallel \bar{a}\langle N \rangle \parallel \bar{c}\langle \llbracket \pi' \rrbracket \rangle$. On a ncessairement $P = \llbracket [N/x]M \rrbracket \parallel \bar{c}\langle \llbracket \pi' \rrbracket \rangle$; 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 (GRAB₂). □

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

Démonstration. Consquence immdiate des Lemmes 2 et 5. \square

Thorme 8. *Pour toute configuration C , nous avons C termine si et seulement si $\llbracket C \rrbracket \xrightarrow{\tau} \star \bar{b}\langle \mathbf{0} \rangle$.*

Démonstration. La machine abstraite s'arrte exactement sur les configurations de la forme $\lambda'x.M \star []$, dont les encodages sont de la forme $(a(x).\llbracket M \rrbracket) \parallel \bar{b}\langle \mathbf{0} \rangle$, 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. \square

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 $\mathbf{n}(\alpha)$ les noms de canaux de α .

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

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 $\Omega \triangleq \nu a.\bar{a}\langle a(x).\bar{a}\langle x \rangle \parallel x \rangle \parallel a(x).\bar{a}\langle x \rangle \parallel x$. Pour tout λ -terme M et toute pile π , on a $\nu a.\nu c.\llbracket M \star \pi \rrbracket \simeq \Omega$ 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 $\nu a.\nu c.\llbracket M \star \pi \rrbracket$ lui est quivalent, puisqu'il ne pourra jamais mettre sur son seul nom libre (b). Inversement, si $M \star \pi$ termine, alors $\nu a.\nu c.\llbracket M \star \pi \rrbracket$ finira par mettre sur b , ce qui le distingue du processus Ω . \square

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 :

$$\begin{aligned} \text{cc} \star M &:: \pi \mapsto M \star k_\pi :: \pi && \text{(CALLCC)} \\ k_\pi \star M &:: \pi' \mapsto M \star \pi && \text{(RESTORE)} \end{aligned}$$

Etant donn un processus P , on dfinit

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

Les deux opérateurs sont alors traduits comme suit :

$$\begin{aligned} \llbracket \mathbf{cc} \rrbracket &\triangleq c(s_0).(s_0 \parallel c(s).a(u).(u \parallel \bar{c}\langle \bar{a}\langle K(s) \rangle \parallel \bar{c}\langle s \rangle \rangle)) \\ \llbracket k_\pi \rrbracket &\triangleq K(\llbracket \pi \rrbracket) \end{aligned}$$

Cet encodage ncessite trois transitions pour simuler les rgles (CALLCC) et (RESTORE) de la KAM avec opérateurs 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.

$$\begin{aligned} \mathbf{cc} \star \pi &\mapsto \mathbf{cc}_1 \star \pi && \text{(CALLCC}_1\text{)} \\ \mathbf{cc}_1 \star M :: \pi &\mapsto \mathbf{cc}_2 \star M :: \pi && \text{(CALLCC}_2\text{)} \\ \mathbf{cc}_2 \star M :: \pi &\mapsto M \star k_\pi :: \pi && \text{(CALLCC}_3\text{)} \\ \\ k_\pi \star \pi' &\mapsto k_\pi^1 \star \pi' && \text{(RESTORE}_1\text{)} \\ k_\pi^1 \star M :: \pi' &\mapsto k_\pi^2 \star M :: \pi' && \text{(RESTORE}_2\text{)} \\ k_\pi^2 \star M :: \pi' &\mapsto M \star \pi && \text{(RESTORE}_3\text{)} \end{aligned}$$

Ces quatre configurations s'encodent comme suit dans HOcore.

$$\begin{aligned} \llbracket \mathbf{cc}_1 \star \pi \rrbracket &\triangleq \llbracket \pi \rrbracket \parallel c(s).a(u).(u \parallel \bar{c}\langle \bar{a}\langle K(s) \rangle \parallel \bar{c}\langle s \rangle \rangle) \\ \llbracket \mathbf{cc}_2 \star M :: \pi \rrbracket &\triangleq \bar{a}\langle \llbracket M \rrbracket \rangle \parallel a(u).(u \parallel \bar{c}\langle \bar{a}\langle K(\llbracket \pi \rrbracket) \rangle \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle \rangle) \\ &= \bar{a}\langle \llbracket M \rrbracket \rangle \parallel a(u).(u \parallel \bar{c}\langle \bar{a}\langle \llbracket k_\pi \rrbracket \rangle \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle \rangle) \\ &= \bar{a}\langle \llbracket M \rrbracket \rangle \parallel a(u).(u \parallel \bar{c}\langle \llbracket k_\pi :: \pi \rrbracket \rangle) \\ \llbracket k_\pi^1 \star \pi' \rrbracket &\triangleq \llbracket \pi' \rrbracket \parallel a(u).c(_).(u \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle) \\ \llbracket k_\pi^2 \star M :: \pi' \rrbracket &\triangleq \bar{c}\langle \llbracket \pi' \rrbracket \rangle \parallel c(_).(\llbracket M \rrbracket \parallel \bar{c}\langle \llbracket \pi \rrbracket \rangle) \\ &= \bar{c}\langle \llbracket \pi' \rrbracket \rangle \parallel c(_).\llbracket M \star \pi \rrbracket \end{aligned}$$

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

$$\begin{aligned} \llbracket \mathbf{cc} \star M :: \pi \rrbracket &\xrightarrow{\tau} \llbracket \mathbf{cc}_1 \star M :: \pi \rrbracket \xrightarrow{\tau} \llbracket \mathbf{cc}_2 \star M :: \pi \rrbracket \xrightarrow{\tau} \llbracket M \star k_\pi :: \pi \rrbracket \\ \llbracket k_\pi \star M :: \pi' \rrbracket &\xrightarrow{\tau} \llbracket k_\pi^1 \star M :: \pi' \rrbracket \xrightarrow{\tau} \llbracket k_\pi^2 \star M :: \pi' \rrbracket \xrightarrow{\tau} \llbracket M \star \pi \rrbracket \end{aligned}$$

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'opérateur \mathbf{cc} effectue d'abord une rception $c(s)$ avant la rception $a(u)$. Ce choix est purement esththique : changer l'ordre des rceptions est tout fait possible, mais ne permet pas d'utiliser $\llbracket k_\pi :: \pi \rrbracket$ dans la traduction de \mathbf{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 $\bar{a}\langle P \rangle$ par $\bar{a}\langle P \rangle.Q$. 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

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

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

$$\frac{}{\bar{a}\langle P \rangle.Q \xrightarrow{\bar{a}\langle P \rangle} Q} \text{OUT}$$

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 $\bar{a}\langle 1 \rangle.\bar{a}\langle \bar{a}\langle 2 \rangle.\bar{a}\langle \bar{a}\langle 3 \rangle.\bar{a}\langle \bar{b}\langle \mathbf{0} \rangle \rangle \rangle \rangle$.

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

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

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

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

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 troisieme 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 sequentialiser 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 simplicité de notre traduction, et le fait qu'elle permette de traiter immédiatement les opérateurs de contrôle, nous laisse espérer qu'elle permettra une meilleure compréhension de l'expressivité du calcul HOcore, et peut-être, terme, résoudre la question de la décidabilité de l'équivalence 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. Gabbriellini, 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.