HABILITATION À DIRIGER LES RECHERCHES

présentée

À L'UNIVERSITÉ PARIS 11

Spécialité : Informatique Fondamentale

par

Hugo HERBELIN

Titre :

C'est maintenant qu'on calcule

au cœur de la dualité

Soutenue le 7 décembre 2005 devant la Commission d'examen composée de

Philippe de Groote	**Rapporteurs**
Jean-Louis Krivine	
Gordon Plotkin	
Stefano Berardi	**Examinateurs**
Andrzej Filinski	
Christine Paulin	

(version soutenance + corrections du 13 décembre 2005)

Table des matières

Introduction

Ce mémoire présente et étudie divers aspects du calcul $\mu\tilde{\mu}$. Ce calcul est à l'intersection de la théorie de la démonstration et de la théorie des programmes. Il constitue le cœur du calcul $\overline{\lambda}\mu\tilde{\mu}$ qui fut mit au point en collaboration avec Pierre-Louis Curien en 2000 [CH00].

Sa structure est à la fois celle du calcul des séquents de Gentzen et celle des machines à environnements, deux notions qui se trouvent ainsi reliées.

Son expressivité logique est celle de la logique classique et son expressivité calculatoire est celle des opérateurs de contrôles, dans la lignée de l'opérateur J de Landin [Lan65a, Lan65b] (simplifié[1] en `escape` chez Reynolds [Rey72] et implanté en Scheme sous la forme d'abord d'un `catch`/`throw` statique[2] [SS75] puis d'un `call-with-current-continuation` [KCR$^+$98]), généralisé en $\mathcal{C}$ par Felleisen *et al* [FFKD86] puis disséqué en μ et [ ] par Parigot [Par92] (le calcul $\lambda\mu$).

Sa syntaxe symétrique se situe dans la lignée du λ-calcul symétrique de Barbanera et Berardi [BB96] et du λ-calcul symétrique de Filinski [Fil89b, Fil89a]). Contrairement au cas du λ-calcul de Barbanera et Berardi, mais pareillement à celui de Filinski, la symétrie exprime une dualité entre les notions standard de terme et de contexte d'évaluation en théorie des programmes.

Sa sémantique est la superposition de l'appel par nom, habituellement considéré en théorie de la démonstration, et de l'appel par valeur, plus souvent considéré en théorie des programmes. De la théorie de l'appel par valeur, telle qu'initiée par Plotkin [Plo75], le calcul $\mu\tilde{\mu}$ apporte une description canonique qui contraste avec la diversité des approches de Moggi [Mog88] (cas intuitionniste) et de Sabry et Felleisen [SF93], Hofmann [Hof95] et Selinger [Sel01] (cas classique).

Au bilan, sur le plan de la théorie de la démonstration, le calcul $\mu\tilde{\mu}$ donne un sens calculatoire au calcul des séquents (symétrique) de Gentzen en l'interprétant comme la superposition de l'appel par nom et de l'appel par valeur. En particulier, il explique pourquoi la traduction du calcul des séquents en déduction naturelle n'est pas injective : le calcul des séquents couvre l'appel par valeur ce que ne couvre pas la déduction naturelle standard qui est une déduction naturelle en appel par nom.

Au bilan, sur le plan de la théorie des programmes, le calcul $\mu\tilde{\mu}$ propose une nouvelle syntaxe internalisant une dualité entre termes et contextes d'évaluation dans la lignée du λ-calcul symétrique de Filinski [Fil89b, Fil89a]. Cette nouvelle syntaxe internalise aussi une dualité entre appel par nom et appel par valeur qui ne s'exprimait alors dans le λ-calcul symétrique de Filinski qu'au niveau de la sémantique dénotationnelle du calcul et chez Selinger [Sel01] qu'au niveau de la sémantique catégorique du calcul $\lambda\mu$.

D'une certaine manière, ce mémoire s'inscrit dans une tradition remontant à Gerhard Gentzen visant à exprimer le sens dans la syntaxe. En particulier, le calcul des séquents et la propriété de la sous-formule en constituent le socle.

[1] Cf par exemple Thielecke [Thi98] pour la connexion entre J et `escape` (et `call-with-current-continuation`).
[2] Par opposition au `catch`/`throw` à liaison dynamique des Lisp (cf [SG93] pour une histoire du Lisp).

Chapitre 1

Correspondance de Curry-Howard, logique classique et calcul des séquents

1.1 Histoire du calcul $\overline{\lambda}\mu\tilde{\mu}$

La correspondance preuves/programmes

Après le constat de la similitude syntaxique entre logique de Hilbert et logique combinatoire typée par Curry à la fin des années 1950 [CFC58], la décennie suivante vit l'observation d'une même similitude entre déduction naturelle et λ-calcul typé par Howard [How80], et, indépendamment, l'exploitation de l'identité structurelle entre preuves et programmes et entre propositions et types par de Bruijn dans le système Automath [dB78].

Extension de la correspondance preuves/programmes à la logique classique

En 1990, Griffin [Gri90] met en évidence que l'opérateur $\mathcal{C}$ de Felleisen-Friedman-Kohlbecker-Duba donne un contenu calculatoire à la logique classique. Se développe alors un intérêt croissant pour l'exploration de nouveaux champs d'applicabilité de la correspondance entre preuves et programmes.

En 1992, Parigot [Par92] met au point le calcul $\lambda\mu$ (ou $\lambda\mu$-calcul par anglicisme) qui permet d'interpréter les preuves de la déduction naturelle classique comme des λ-termes, avec la réduction simulant la normalisation des preuves.

Extension de la correspondance preuves/programmes au calcul des séquents

En 1994, nous donnons une interprétation de la règle d'introduction gauche du calcul des séquents LK de Gentzen [Gen35] comme constructeur de liste d'arguments [Her95]. Cette nouvelle construction permet de voir le calcul des séquents à son tour comme un λ-calcul. Cependant, elle ne couvre qu'un fragment restreint de LK, à savoir la restriction LKT (et LJT pour la variante intuitionniste) définie par Danos, Joinet et Schellinx [DJS95] (cf aussi Mints [Min96] pour un calcul équivalent à LJT).

Un des problèmes pour interpréter calculatoirement le calcul des séquents est que toute tentative de réduction des preuves conduit à une rupture arbitraire de symétrie (cf la question du « dilemme » chez Danos, Joinet et Schellinx [DJS97] et l'annexe B de Girard-Lafont-Taylor [GLT89]). En 1991, dans la foulée de la prise de conscience d'un contenu calculatoire à la logique classique, Girard développe un argument sémantique pour « autoriser » à briser la symétrie de l'élimination des coupures dans le calcul des séquents : les formules négatives (connecteurs $⅋$, & et ? de la logique linéaire) étant inversibles, elle se feront dupliquer/effacer par les formules positives (connecteurs $\otimes$, $\oplus$ et ! de la logique linéaire). Cette approche « sémantique » conduisit au calcul des séquents LC [Gir91], dont la syntaxe en tant que calcul de termes a été caractérisée par Murthy dans un calcul de continuations [Mur92].

L'appel par valeur

Du fait de son absence de symétrie gauche/droite, la syntaxe de la déduction naturelle classique cachait la conscience d'un « dilemme » incontournable. C'est tout naturellement que Parigot privilégia (implicitement) la restriction à l'appel par nom.

En 1995, conjointement à la définition de LKT, Danos, Joinet et Schellinx définirent un calcul dual de LKT nommé LKQ (T = tête, Q = queue). Il était plus ou moins implicite à l'époque que l'interprétation intuitionniste de $A \to B$ en logique linéaire comme $!A \multimap B$ correspondait à l'appel par nom tandis que l'interprétation comme $!A \multimap !B$ correspondait à l'appel par valeur.

La genèse du calcul $\overline{\lambda}\mu\tilde{\mu}$

En 1998, je découvris l'interprétation de l'appel par valeur en termes de jeux de Honda et Yoshida [HY97]. Je fus frappé par la correspondance exacte entre leurs formes normales du λ-calcul en appel par valeur (que je devais réaliser bien plus tard être celles du calcul « computationnel[1] » de Moggi [Mog88]) et les preuves normales de LJQ (c'est-à-dire de la restriction intuitionniste de LKQ) : la dualité T/Q était bien une dualité nom/valeur.

En 1999, suite à un exposé de Vincent Danos sur, je crois, l'interprétation de LJQ par des termes obtenus par passage de continuation à partir du λ-calcul par valeur, je mis au point une première interprétation directe d'une variante de LKQ, à la suite de laquelle suivirent de nombreuses autres variantes pour LKQ et LKT. Je ne trouvais pas d'arguments satisfaisants pour préférer une version à l'autre. En particulier, j'étais motivé par modéliser au plus près de l'usage courant les variantes en appel par nom et en appel par valeur du calcul $\lambda\mu$ et je ne me départais pas de l'idée d'avoir toujours au moins une conclusion active.

J'exposais alors mes variantes et mes indécisions à Pierre-Louis Curien qui posa alors la question d'une « dualité » syntaxique, incitant par là à préférer la dualité à la fidélité envers l'usage. Cela correspondait d'ailleurs à une période où Selinger [Sel01] mettait au point une parfaite dualité catégorique entre appel par nom et appel par valeur, avec application à la modélisation des versions appel par nom et appel par valeur du calcul $\lambda\mu$, et une preuve de leur isomorphisme, concrétisant ainsi de manière explicite l'observation précoce de la dualité sémantique entre appel par nom et appel par valeur par Filinski [Fil89a, Fil89b]. Dans la semaine qui suivit ma discussion avec Pierre-Louis, chacun d'entre nous mit au point indépendamment une première version d'un calcul strictement dual qui allait devenir le calcul $\overline{\lambda}\mu\tilde{\mu}$.

Une autre de mes lignes directrices était la fidélité à la structure du calcul des séquents et notamment au principe « coupure = rédex ». Pierre-Louis, dont la culture était plus λ-calcul et sémantique catégorique, proposa un calcul à deux axiomes dont la règle de coupure n'était plus forcément « destructrice ». Incidemment, c'est grâce à ce choix que le calcul $\overline{\lambda}\mu\tilde{\mu}$ accepte une présentation arborescente avec contexte implicite (comme en déduction naturelle).

L'apport du calcul $\overline{\lambda}\mu\tilde{\mu}$

En résumé, on peut estimer que le calcul $\overline{\lambda}\mu\tilde{\mu}$ aura innové sur les points suivants :

- développement d'une syntaxe originale de λ-calcul exprimant de manière syntaxique :
  - « la » dualité entre appel par nom et appel par valeur,
  - « la » dualité entre termes et contextes d'évaluation ;
- mise en évidence d'un sous-système autonome et symétrique $\mu\tilde{\mu}$ gérant la substitution et le dilemme entre appel par nom et appel par valeur ;
- une décomposition du λ-calcul en ce sous-système et une paire de constructeurs, abstraction et application, interagissant en se détruisant mutuellement ;
- une modularité permettant l'ajout de nouvelles constructions constituées d'un constructeur de terme et d'un constructeur de contexte d'évaluation interagissant en se détruisant mutuellement ;
- l'attribution d'un sens calculatoire à la totalité des preuves du calcul des séquents ;
- la possibilité de représenter les preuves du calcul des séquents sous forme arborescente avec contexte implicite, comme pour la déduction naturelle ;
- un système de réduction avec rédex actif en tête, à la manière des machines de réduction abstraites.

[1] Difficile ici de traduire littéralement l'anglais « computational calculus » (ou alors par quelque chose comme « système calculatoire » ou « calcul d'exécutions » ?).

Travaux connexes

Rétrospectivement, le $\overline{\lambda}\mu\tilde{\mu}$ est très proche du λ-calcul symétrique de Barbanera et Berardi [BB96] qui peut être vu comme la variante « additive » (sans abstraction ni application) du calcul $\overline{\lambda}\mu\tilde{\mu}$ qu'on aurait quotientée en identifiant termes et contextes d'évaluation et appel par nom et appel par valeur. Autrement dit, le $\overline{\lambda}\mu\tilde{\mu}$ ré-extrait du λ-calcul symétrique un dualité calculatoire (en termes standards de la théorie de la programmation) dont la perspective avait été repliée par la mise en œuvre radicale de la symétrie syntaxique.

Barbanera et Berardi donnèrent une preuve intéressante de normalisation non-déterministe de leur calcul. Urban et Bierman [UB01] exploitèrent l'idée de cette preuve pour donner une preuve de normalisation non-déterministe à grand pas[2] du calcul des séquents LK.

Lengrand [Len03] reprit la formulation de LK de Urban et Bierman. Il donna une présentation de la restriction à l'implication (mais non typée) comme un λ-calcul non déterministe de nom $\lambda\xi$-calcul qu'il compara au $\overline{\lambda}\mu\tilde{\mu}$.

Indépendamment, Parigot [Par00] symétrisa le calcul $\lambda\mu$ conduisant à une variante « déduction naturelle » et non déterministe du λ-calcul symétrique avec abstraction et application (le μ correspondant au λ de Barbanera et Berardi).

Wadler [Wad03] proposa une variante du calcul $\overline{\lambda}\mu\tilde{\mu}$ basée sur une conjonction et une disjonction additives plutôt que sur les constructeurs abstraction et application de l'implication. Ces derniers peuvent en retour être simulés à partir des connecteurs symétriques. Toutefois, la simulation utilise les aspects classiques du calcul de manière cruciale et n'est pas compatible avec la restriction intuitionniste.

1.2 Du calcul des séquents aux calculs $\overline{\lambda}\mu\tilde{\mu}$ et $LK_{\mu\tilde{\mu}}$

La structure du calcul $\overline{\lambda}\mu\tilde{\mu}$ est celle des preuves du calcul des séquents et nous appellerons $LK_{\mu\tilde{\mu}}^{\rightarrow}$ son système de types simples (on abrégera parfois $LK_{\mu\tilde{\mu}}^{\rightarrow}$ en $LK_{\mu\tilde{\mu}}$ lorsque la question de quels connecteurs sont ou non présents dans le système n'importe pas).

Considérons par exemple le calcul des séquents classique restreint à l'implication, exprimé avec des règles d'échange implicites (grâce à une définition des contextes de formules comme multi-ensembles), des règles d'affaiblissement déportées vers les axiomes (et intégrées à ceux-ci) et des règles de contraction explicites.

$$Ax \ \frac{}{\Gamma, A \vdash A, \Delta}$$

$$\rightarrow_L \ \frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta} \qquad \rightarrow_R \ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}$$

$$Cont_L \ \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad Cont_R \ \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

$$Cut \ \frac{\Gamma \vdash A \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}$$

Une première étape pour aller du calcul des séquents au calcul $LK_{\mu\tilde{\mu}}^{\rightarrow}$ consiste à distinguer trois types de séquents. Le premier type de séquents aura une formule distinguée à droite (à la manière par exemple du calcul $\lambda\mu$, mais en fait implicitement à la manière de tout calcul intuitionniste), le second une formule distinguée à gauche (à la manière du calcul LKT de Danos, Joinet et Schellinx [DJS95]), et le troisième aucune formule distinguée (à la manière des séquents interprétant les termes nommés dans le calcul $\lambda\mu$). Les séquents ayant une formule distinguée seront dits « avec bénitier » et la formule distinguée sera dite « dans le bénitier », suivant une terminologie initiée par Girard [Gir91]. On utilise un point-virgule pour signaler un éventuel bénitier.

Les principes de base de $LK_{\mu\tilde{\mu}}$ et de ses extensions sont alors les suivants :

[2] C'est-à-dire basée sur une substitution immédiate d'un côté de la coupure vers les axiomes introduisant la formule coupée.

– toute règle d'introduction construira un séquent avec bénitier ;
– la règle de coupure met en relation une formule dans le bénitier à droite avec la même formule dans le bénitier à gauche et produit un séquent sans formule distinguée ;
– les axiomes sont assimilés à des règles d'introduction.

Sur ces bases, on constate par symétrie qu'il faut deux règles d'axiomes. On obtient alors :

$$Ax_L \; \frac{}{\Gamma; A \vdash A, \Delta} \qquad Ax_R \; \frac{}{\Gamma, A \vdash A; \Delta}$$

$$\to_L \; \frac{\Gamma \vdash A; \Delta \qquad \Gamma; B \vdash \Delta}{\Gamma; A \to B \vdash \Delta} \qquad \to_R \; \frac{\Gamma, A \vdash B; \Delta}{\Gamma \vdash A \to B; \Delta}$$

$$Cut \; \frac{\Gamma \vdash A; \Delta \quad \Gamma; A \vdash \Delta}{\Gamma \vdash \Delta}$$

où il reste à interpréter les règles de contraction.

On constate en premier lieu une part d'arbitraire concernant les prémisses des règles d'introduction. Typiquement, pour la règle d'introduction droite de l'implication, la formule distinguée aurait pu être à gauche. C'est un phénomène qui se reproduira pour d'autres connecteurs et nous verrons que selon les besoins, l'un ou l'autre des choix à disposition pourront être mis en oeuvre.

On constate en second lieu qu'aucune règle ne permet de poursuivre la construction d'une preuve se terminant par une coupure.

Pour cela, on ajoute deux nouvelles règles de **focalisation** permettant de distinguer une formule, soit à droite, soit à gauche, dans un séquent qui n'a aucune formule distinguée. À droite, la focalisation correspond à la règle μ du calcul $\lambda\mu$. Par symétrie de notation, on appellera $\tilde{\mu}$ la règle de focalisation à gauche.

$$\tilde{\mu} \; \frac{\Gamma, A \vdash \Delta}{\Gamma; A \vdash \Delta} \qquad \mu \; \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A; \Delta}$$

Reste la question des règles de contraction. Là, le choix est de renoncer à l'élimination des coupures et de simuler la contraction à partir de la coupure avec une règle axiome (avec la règle axiome droite pour la contraction à gauche, avec la règle axiome gauche pour la contraction à droite).

$$Cut \; \frac{Ax_R \; \dfrac{}{\Gamma, A \vdash A; \Delta} \qquad \Gamma, A; A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad Cut \; \frac{\Gamma \vdash A; A, \Delta \qquad Ax_L \; \dfrac{}{\Gamma; A \vdash A, \Delta}}{\Gamma \vdash A, \Delta}$$

Nous appelons $LK_{\mu\tilde{\mu}/}$ le calcul de base constitué des règles Ax_L, Ax_R, $\tilde{\mu}$, μ et Cut pour lequel les contextes sont des multi-ensembles de formules. L'ajout de $\to_L$ et $\to_R$ donne $LK^{\to}_{\mu\tilde{\mu}/}$.

Les calculs $LK_{\mu\tilde{\mu}}$ et $LK^{\to}_{\mu\tilde{\mu}}$ seront décrits ultérieurement (en sections 2.15 et 3.2.9). Ils diffèrent de $LK_{\mu\tilde{\mu}/}$ et $LK^{\to}_{\mu\tilde{\mu}/}$ par la représentation des contextes de formules qui ne sont plus des contextes de formules au sens strict mais des contextes de formules nommées, de telle sorte que deux occurrences distinctes de la même formule soient distinguables (condition nécessaire pour interpréter les preuves calculatoirement de manière non ambiguë). En particulier, $LK_{\mu\tilde{\mu}/}$ est le quotient de $LK_{\mu\tilde{\mu}}$ obtenu en identifiant les formules identiques de noms différents (on appelle ce quotient le quotient hypothético-amalgamant).

1.3 Représentation arborescente de $LK_{\mu\tilde{\mu}}$

L'absence de règle de contraction, tant à droite qu'à gauche, fait qu'une représentation des calculs $LK_{\mu\tilde{\mu}}$ et $LK_{\mu\tilde{\mu}/}$ sous forme arborescente avec contexte implicite est possible. À la différence de la déduction naturelle, la représentation arborescente est basée sur trois types de dérivation : les dérivations de conclusion, notées $\vdash A$, les dérivations d'hypothèse, notées $A \vdash$ et les dérivations « contradictoires », notées $\vdash$.

$$
\rightarrow_L \quad \frac{\vdash A \qquad B \vdash}{A \rightarrow B \vdash}
\qquad\qquad
\rightarrow_R \quad \frac{\begin{array}{c}[\vdash A] \\ \vdots \\ \vdash B\end{array}}{\vdash A \rightarrow B}
$$

$$
\tilde{\mu} \quad \frac{\begin{array}{c}[\vdash A] \\ \vdots \\ \vdash\end{array}}{A \vdash}
\qquad\qquad
\mu \quad \frac{\begin{array}{c}[A \vdash] \\ \vdots \\ \vdash\end{array}}{\vdash A}
$$

$$
Cut \quad \frac{\vdash A \qquad A \vdash}{\vdash}
$$

Chapitre 2

Le sous-système $\mu\tilde{\mu}$

On oublie le typage et on se concentre sur la structure calculatoire du calcul des séquents précédent. Cette structure est suffisamment riche pour qu'on puisse aussi laisser de côté l'implication. Plus précisément, nous allons raisonner de manière abstraite en supposant donné un ensemble de constructeurs dont la dynamique sera étudiée dans le chapitre 3. En particulier, on retrouvera l'implication en section 3.2. Ce cœur calculatoire du calcul des séquents, tel que nous l'avons analysé, est appelé sous-système $\mu\tilde{\mu}$. Nous le déclinerons en trois variantes. Une version non-déterministe symétrique et deux versions asymétriques duales l'une de l'autre qui s'interpréteront comme des restrictions appel par nom et appel par valeur de la version non-déterministe.

2.1 Conventions liminaires

Dans la suite, on adopte les conventions d'écriture suivantes pour les relations de réduction définies dans ce chapitre. Une réduction $\rightarrow$ sans autre annotation désigne une réduction contextuelle, c'est-à-dire compatible avec les constructions des langages considérés. La notation $\stackrel{h}{\rightarrow}$ désigne une réduction qui modifie la tête d'une expression sans inclure le passage au contexte. Si $\rightarrow$ est une relation de réduction, sa clôture réflexive-transitive est notée $\stackrel{*}{\rightarrow}$.

La notation $::=$ désigne une définition inductive (définition d'un type algébrique par une définition de grammaire). Le symbole $\triangleq$ désigne une définition (une abréviation). Quant à la notation $=$, elle désigne en général l'égalité (la conversion) dans un des calculs considérés dans ce mémoire. En général, elle est accompagnée d'une annotation explicitant quelles règles de réduction ou de conversion sont considérées. Autrement dit, $=$ désigne la clôture symétrique et transitive de $\rightarrow$.

2.2 Syntaxe brute

Les objets associés aux séquents avec bénitier à droite sont appelés **termes**. Ceux associés aux séquents avec bénitier à gauche sont appelés **contextes d'évaluation** (ou plus simplement **contextes** en absence d'ambiguïté). Parmi les objets associés à des séquents avec bénitier, on distingue ceux provenant des règles d'introduction de connecteurs : on les appelle **constructeurs d'expressions linéaires**. S'il construisent à droite, on les appelle constructeurs de valeurs, et si à gauche, constructeurs de contextes d'évaluation linéaire. Enfin, les objets associés aux séquents sans bénitier sont appelés **commandes** (ou parfois **exécutables**, ou encore **états**, ou encore **interactions**).

Pour interpréter les règles d'axiome et les règles de focalisation, il est nécessaire d'associer des noms aux formules. On associe aux hypothèses non distinguées des noms de variables parcourant un ensemble infini $\mathcal{X}$ dont les éléments sont appelés **variables de terme** et sont représentés avec les lettres minuscules x, y, z, etc. De la même manière, on associe aux conclusions non distinguées des noms de variables parcourant un ensemble infini $\mathcal{A}$ dont les éléments sont appelés **variables de contexte d'évaluation** et sont représentés avec les lettres grecques minuscules α, β, γ, etc. comme les variables de conclusion du calcul $\lambda\mu$.

La description du sous-système $\mu\tilde{\mu}$ prend en paramètre la la catégorie des **constructeurs de valeurs**, dont les éléments sont représentés par la lettre indexée V_c et ses variations, et celle des **constructeurs**

de contextes d'évaluation linéaires, dont les éléments sont représentés par la lettre indexée E_c et ses variations. Ces catégories peuvent faire référence aux autres catégories du calcul. En particulier, les éléments de ces catégories peuvent contenir des variables et la substitution de ces variables est supposée définie. Une **valeur** est soit une variable de terme, soit un constructeur de valeurs. Un **contexte d'évaluation linéaire** est soit une variable de contexte d'évaluation, soit un constructeur de contexte d'évaluation. Valeurs et contextes d'évaluation linéaires rentrent plus généralement dans la catégorie des expressions **linéaires**. En particulier, la terminologie « valeur » est un raccourci pour « terme linéaire ».

La syntaxe (paramétrique) du sous-système $\mu\tilde{\mu}$ est la suivante

Syntaxe brute

Commandes	c	$::=$	$\langle v \| e\rangle$
Constructeurs de valeurs	V_c	$::=$	$\ldots$
Valeurs	V	$::=$	$x \mid V_c$
Termes	v	$::=$	$\mu\alpha.c \mid V$
Constructeurs de contextes d'évaluation linéaires	E_c	$::=$	$\ldots$
Contextes d'évaluation linéaires	E	$::=$	$\alpha \mid E_c$
Contextes d'évaluation	e	$::=$	$\tilde{\mu}x.c \mid E$

Les variables interprètent les règles d'axiome. L'expression $\langle v\|e\rangle$ représente la règle de coupure. Calculatoirement, elle s'interprète comme l'instanciation du contexte d'évaluation e par le terme v. Cette instanciation correspond à une interaction qui peut déclencher des réductions. Les expressions de la forme $\mu\alpha.c$ et $\tilde{\mu}x.c$ sont des lieurs.

Dans une commande, un terme de la forme $\mu\alpha.c$ capture son contexte d'évaluation, et le substitue à chaque occurrence de α dans c. Il joue le même rôle que μ dans le calcul $\lambda\mu$ de Parigot à ceci près que la substitution se fait en une étape dans le système $\mu\tilde{\mu}$ alors qu'elle se fait par propagation successive des constructions élémentaires de contexte dans le calcul $\lambda\mu$.

L'expression $\tilde{\mu}x.c$ est une notation (duale de la notation $\mu\alpha.c$) pour un contexte d'évaluation de la forme **let** $x = \Box$ **in** c. L'opération de substitution associée correspond à la contraction du `let-in`.

La différence principale entre les contextes d'évaluation en $\lambda\mu$-calcul et ceux du système $\mu\tilde{\mu}$ est que les premiers sont des méta-notations alors que les seconds sont des constructions algébriques. Par exemple, le contexte d'évaluation α du système $\mu\tilde{\mu}$ correspond au contexte $[\alpha](\Box)$ du $\lambda\mu$-calcul. De même, la notation $\langle v\|e\rangle$ correspond au niveau du $\lambda\mu$-calcul à l'instanciation $e[v]$ du contexte e par v.

L'ensemble des **variables libres** d'une expression est défini de manière standard. Il y a juste à distinguer les variables libres de contexte FV_e et les variables libres de terme FV_v.

$$\begin{array}{lcl@{\qquad\qquad}lcl}
FV_e(x) & \triangleq & \emptyset & FV_v(x) & \triangleq & \{x\} \\
FV_e(\alpha) & \triangleq & \{\alpha\} & FV_v(\alpha) & \triangleq & \emptyset \\
FV_e(\mu\alpha.c) & \triangleq & FV_e(c) \setminus \{\alpha\} & FV_v(\mu\alpha.c) & \triangleq & FV_v(c) \\
FV_e(\tilde{\mu}x.c) & \triangleq & FV_e(c) & FV_v(\tilde{\mu}x.c) & \triangleq & FV_v(c) \setminus \{x\} \\
FV_e(\langle v\|e\rangle) & \triangleq & FV_e(v) \cup FV_e(e) & FV_v(\langle v\|e\rangle) & \triangleq & FV_v(v) \cup FV_v(e)
\end{array}$$

$$\begin{array}{lcl}
FV(v) & \triangleq & FV_v(v) \cup FV_e(v) \\
FV(e) & \triangleq & FV_v(e) \cup FV_e(e) \\
FV(c) & \triangleq & FV_v(c) \cup FV_e(c)
\end{array}$$

On utilisera généralement le caractère _ (souligné) pour le nom d'une variable n'apparaissant pas dans l'expression dans laquelle elle est liée. Par exemple $\mu\alpha.\langle x\|\beta\rangle$ sera noté $\mu\_.\langle x\|\beta\rangle$. Pour faciliter la lecture, des expressions contenant des commandes imbriquées, on surlignera les commandes qui figurent comme sous-expression d'autres commandes. Par exemple,

$$\langle \mu\alpha.\langle y_1\|\beta_1\rangle \| \tilde{\mu}x.\langle y_2\|\beta_2\rangle\rangle$$

sera écrit

$$\langle \mu\alpha.\langle\overline{y_1\|\beta_1}\rangle \| \tilde{\mu}x.\langle\overline{y_2\|\beta_2}\rangle\rangle\,.$$

Comme déjà évoqué, les **formes normales** du sous-système $\mu\tilde{\mu}$ ne sont pas des formes sans coupure. Ce sont des formes dont les seules instances de coupure mettent en jeu d'un côté une variable et de l'autre une expression linéaire (valeur ou contexte d'évaluation linéaire). Autrement dit, les formes normales sont celles dont les sous-commandes ont la forme $\langle x\|E_c\rangle$, ou $\langle V_c\|\alpha\rangle$, ou $\langle x\|\alpha\rangle$.

2.3 Réduction non-déterministe

On s'intéresse d'abord à une version symétrique des règles de réduction du sous-système $\mu\tilde{\mu}$:

Réduction non déterministe

$$\begin{array}{lccc} (\mu) & \langle \mu\alpha.c \| e \rangle & \xrightarrow{h} & c[\alpha \leftarrow e] \\ (\tilde{\mu}) & \langle v \| \tilde{\mu}x.c \rangle & \xrightarrow{h} & c[x \leftarrow v] \end{array}$$

où $c[\alpha \leftarrow e]$ et $c[x \leftarrow v]$ (et plus généralement $v'[\alpha \leftarrow e]$, $v'[x \leftarrow v]$ et $e'[\alpha \leftarrow e]$ et $e'[x \leftarrow v]$) désignent respectivement la substitution sans capture de α par e dans c et de x par v dans c.

De manière évidente, les règles (μ) et $(\tilde{\mu})$ ne définissent pas un système confluent. Par exemple, on a

$$\begin{array}{ccc} & \langle \mu\alpha.\overline{\langle y_1 \| \beta_1 \rangle} \| \tilde{\mu}x.\overline{\langle y_2 \| \beta_2 \rangle} \rangle & \\ {}^{(\mu)}\swarrow & & \searrow^{(\tilde{\mu})} \\ \langle y_1 \| \beta_1 \rangle & & \langle y_2 \| \beta_2 \rangle \end{array}$$

On peut casser le non-déterminisme en imposant des restrictions asymétriques. On appelle **appel par nom**, en abrégé **CBN** (« Call-By-Name »), le système de réduction confluent obtenu en donnant systématiquement la préférence à la règle $\tilde{\mu}$ en cas de conflit. Symétriquement, on appelle **appel par valeur**, abrégé **CBV** (« Call-By-Value »), le système de réduction confluent obtenu en donnant systématiquement la préférence à la règle μ.

La non-symétrie syntaxique des terminologies « appel par nom » et « appel par valeur » est historique. Ces terminologies font référence à la manière avec laquelle une fonction se comporte en λ-calcul vis à vis de son argument. Transplantées dans le cadre du sous-système $\mu\tilde{\mu}$, elles expriment comment un contexte de la forme $\tilde{\mu}x.c$ se comporte vis à vis de « son » argument. L'appel par valeur signifie que le contexte exige que l'argument soit une valeur avant de le substituer à x tandis que l'appel par nom signifie que le contexte lie son argument au nom x tel que sans chercher à l'évaluer (l'évaluation se fera dans les sous-termes faisant référence à x). On verra en sections 3.13.3 et 3.13.4 que cette terminologie coïncide bien avec la terminologie usuelle du λ-calcul.

Les restrictions appel par nom et appel par valeur seront décrites plus en détail dans les prochaines sections. Dans l'immédiat, nous nous intéresserons à la terminaison de la réduction du sous-système $\mu\tilde{\mu}$ non-déterminisme.

2.4 Terminaison de la réduction non-déterministe

Le sous-système $\mu\tilde{\mu}$ non déterministe est équivalent au fragment (typé) sans conjonction ni disjonction du λ-calcul symétrique de Barbanera et Berardi. Leur preuve de terminaison par réductibilité [BB96] s'applique (cf Polonovski [Pol03] pour une preuve par réductibilité exprimée directement dans la syntaxe du calcul $\overline{\lambda}\mu\tilde{\mu}$).

Proposition 1 *Le système de réduction $\mu\tilde{\mu}$ non déterministe est fortement normalisant.*

Bien que fortement normalisant, le sous-système non-déterministe est irrémédiablement non confluent et nous nous intéressons dans la suite aux restrictions appel par nom et appel par valeur.

2.5 Appel par nom

En appel par nom, la priorité de $(\tilde{\mu})$ sur (μ) implique que les variables de contexte ne sont instanciées que par des constructeurs de contexte d'évaluation linéaire ou par des variables, tandis que les variables de terme peuvent être substituées par n'importe quel genre de terme. Cela justifie de fusionner les catégories V et v.

Syntaxe du sous-système $\mu\tilde{\mu}$ restreint à l'appel par nom

Commandes	c	$::=$	$\langle v \| e\rangle$
Constructeurs de valeurs	V_c	$::=$	$\ldots$
Termes	v	$::=$	$\mu\alpha.c \mid x \mid V_c$
Constructeurs de contextes d'évaluation linéaires	E_c	$::=$	$\ldots$
Contextes d'évaluation linéaires	E	$::=$	$\alpha \mid E_c$
Contextes d'évaluation	e	$::=$	$\tilde{\mu}x.c \mid E$

L'appel par nom se caractérise alors par le système (confluent)

Réduction en appel par nom

$$\begin{array}{llll} (\mu_n) & \langle \mu\alpha.c \| E\rangle & \stackrel{h}{\rightarrow}_n & c[\alpha \leftarrow E] \\ (\tilde{\mu}) & \langle v \| \tilde{\mu}x.c\rangle & \stackrel{h}{\rightarrow}_n & c[x \leftarrow v] \end{array}$$

On appelle sous-système $\mu_n\tilde{\mu}$ le calcul correspondant. Une manière formelle de prouver la confluence est de plonger la restriction CBN dans le cadre générique des systèmes de réécriture d'ordre supérieur (HRS de Nipkow [Nip91]) pour lequel l'orthogonalité (c'est-à-dire linéarité à gauche et absence de paire critique) garantit la confluence.

Proposition 2 *La restriction CBN est confluente*

Preuve: L'interprétation CBN est la suivante :

$$\begin{array}{lll@{\qquad\qquad}lll} \alpha^* & \triangleq & \mathsf{LinCtx}\ \alpha & x^* & \triangleq & x \\ E_c^* & \triangleq & \mathsf{LinCtx}\ E_c & V_c^* & \triangleq & \mathsf{ValCstr}\ V_c \\ (\tilde{\mu}x.c)^* & \triangleq & \mathsf{MuTilde}\ x.c^* & (\mu\alpha.c)^* & \triangleq & \mathsf{Mu}\ \alpha.c^* \end{array}$$
$$\langle v \| e\rangle^* \triangleq \mathsf{Cut}\ v^*\ e^*$$

et le système de réduction orthogonal est

$$\begin{array}{lll} \mathsf{Cut}\ (\mathsf{Mu}\ \alpha.c(\alpha))\ (\mathsf{LinCtx}\ E) & \stackrel{*}{\rightarrow}_n & c(E) \\ \mathsf{Cut}\ v\ (\mathsf{MuTilde}\ x.c(x)) & \stackrel{*}{\rightarrow}_n & c(v) \end{array}$$

On vérifie en particulier que $c \rightarrow_n c'$ ssi $c^* \stackrel{*}{\rightarrow}_n c'^*$. ■

2.6 Appel par valeur

Symétriquement, les variables de terme ne sont instanciées en appel par valeur que par des constructeurs de valeurs ou des variables, ce qui justifie de fusionner les catégories E et e.

Syntaxe du sous-système $\mu\tilde{\mu}$ restreint à l'appel par valeur

Commandes	c	$::=$	$\langle v \| e\rangle$
Constructeurs de valeurs	V_c	$::=$	$\ldots$
Valeurs	V	$::=$	$x \mid V_c$
Termes	v	$::=$	$\mu\alpha.c \mid V$
Constructeurs de contextes d'évaluation linéaires	E_c	$::=$	$\ldots$
Contextes d'évaluation	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid E_c$

La réduction en appel par valeur se caractérise alors par le système (confluent)

Réduction en appel par valeur

$$\begin{array}{llll} (\mu) & \langle \mu\alpha.c \| e\rangle & \stackrel{h}{\rightarrow}_v & c[\alpha \leftarrow e] \\ (\tilde{\mu}_v) & \langle V \| \tilde{\mu}x.c\rangle & \stackrel{h}{\rightarrow}_v & c[x \leftarrow V] \end{array}$$

On appelle sous-système $\mu\tilde{\mu}_v$ le calcul correspondant. Symétriquement au cas CBN, on a

Proposition 3 *La restriction CBV est confluente*

2.7 Propriétés de permutation des lieurs

Les lieurs peuvent être permutés à certaines conditions. Certaines permutations sont valides tant en appel par nom qu'en appel par valeur tandis que d'autres ne sont valides que dans l'une des restrictions. On note $=_{nv}$ pour une propriété valide à la fois en appel par nom et en appel par valeur. On a

Proposition 4 *[Permutation des lieurs] Soient x, y, α et β des variables non libres dans e, e', E, v, v' et V. Les propriétés suivantes sont vraies à la fois en appel par nom et en appel par valeur :*

$$\begin{array}{lll} (\pi_{ve}) & \langle\mu\alpha.\overline{\langle v\|\tilde{\mu}x.c\rangle}\|e\rangle & =_{nv} \quad \langle v\|\tilde{\mu}x.\overline{\langle\mu\alpha.c\|e\rangle}\rangle \\ (\pi_V) & \langle v\|\tilde{\mu}x.\overline{\langle V\|\tilde{\mu}y.c\rangle}\rangle & =_{nv} \quad \langle V\|\tilde{\mu}y.\overline{\langle v\|\tilde{\mu}x.c\rangle}\rangle \\ (\pi_E) & \langle\mu\alpha.\overline{\langle\mu\beta.c\|E\rangle}\|e\rangle & =_{nv} \quad \langle\mu\beta.\overline{\langle\mu\alpha.c\|e\rangle}\|E\rangle \end{array}$$

La propriété suivante n'est en général vraie qu'en appel par nom :

$$(\pi_v) \qquad \langle v\|\tilde{\mu}x.\overline{\langle v'\|\tilde{\mu}y.c\rangle}\rangle \quad =_n \quad \langle v'\|\tilde{\mu}y.\overline{\langle v\|\tilde{\mu}x.c\rangle}\rangle$$

La propriété suivante n'est en général vraie qu'en appel par valeur :

$$(\pi_e) \qquad \langle\mu\alpha.\overline{\langle\mu\beta.c\|e'\rangle}\|e\rangle \quad =_v \quad \langle\mu\beta.\overline{\langle\mu\alpha.c\|e\rangle}\|e'\rangle$$

Preuve: L'équation (π_{ve}) est valide car les deux côtés se réduisent vers $\langle\mu\alpha.c[x \leftarrow v]\|e\rangle$. L'équation (π_V) est valide car les deux côtés se réduisent vers $\langle v\|\tilde{\mu}x.c[y \leftarrow V]\rangle$. L'équation (π_E) est valide car les deux côtés se réduisent vers $\langle\mu\alpha.c[\beta \leftarrow E]\|e\rangle$. L'équation (π_v) est valide car les deux côtés se réduisent en CBN vers $c[y \leftarrow v'][x \leftarrow v]$. En prenant $e \triangleq \tilde{\mu}x.\langle x_0\|\alpha_0\rangle$ et $e' \triangleq \tilde{\mu}x.\langle x_1\|\alpha_1\rangle$, on observe que (π_e) n'est pas valide en général pour l'appel par nom. Que (π_e) est valide en appel par valeur mais ne l'est pas forcément en appel par nom s'obtient par dualité. ■

2.8 Substitutions CBN et CBV

Remarquons que l'appel par nom est stable par substitution des variables de contexte par des contextes linéaires et par substitution des variables de terme par des termes quelconques. Donnons quelques définitions. Une **substitution de termes** est une fonction d'un ensemble de variables de terme vers l'ensemble des termes. Une **substitution de contextes d'évaluation** est une fonction d'un ensemble de variables de contexte d'évaluation vers l'ensemble des contextes d'évaluation. Une **substitution** au sens général est la paire d'une substitution de termes et d'une substitution de contextes d'évaluation. Une substitution est dite **substitution appel par nom** (ou substitution CBN) si elle associe des contextes d'évaluation linéaires aux variables de contexte d'évaluation. Elle est dite **substitution appel par valeur** (ou substitution CBV) si elle associe des valeurs aux variables de terme. La terminologie s'appliquera aussi aux simples substitutions de termes (qui peuvent donc être contraintes à être appel par valeur) et aux substitutions de contextes d'évaluation (qui peuvent être contraintes à être appel par nom). On note $c[\sigma]$ (et plus généralement $v[\sigma]$ et $e[\sigma]$) pour la substitution simultanée dans c des variables de σ par leur image. On a :

Proposition 5
- *Si σ est une substitution CBN alors $c \rightarrow_n c'$ implique $c[\sigma] \rightarrow_n c'[\sigma]$.*
- *Si σ est une substitution CBV alors $c \rightarrow_v c'$ implique $c[\sigma] \rightarrow_v c'[\sigma]$.*

2.9 Principe de dualité

Le sous-système $\mu\tilde{\mu}$ a une profonde symétrie interne. Soit $x \mapsto \alpha_x$ une bijection de $\mathcal{X}$ vers $\mathcal{A}$. On note $\alpha \mapsto x_\alpha$ pour la bijection réciproque. La dualité interne à $\mu\tilde{\mu}$ est définie par :

$$\begin{array}{lll} \langle v\|e\rangle^\circ & \triangleq & \langle e^\circ\|v^\circ\rangle \\ x^\circ & \triangleq & \alpha_x \\ \alpha^\circ & \triangleq & x_\alpha \\ (\mu\alpha.c)^\circ & \triangleq & \tilde{\mu}x_\alpha.c^\circ \\ (\tilde{\mu}x.c)^\circ & \triangleq & \mu\alpha_x.c^\circ \end{array}$$

Les règles de réduction μ et $\tilde{\mu}$ sont duales. La dualité (une involution) échange les règles de réduction par valeur et par nom.

Toute proposition ou opération qui mentionne certains des concepts suivants pris au choix dans la colonne de gauche ou de droite admet un énoncé dual équivalent par substitution avec le concept dual se trouvant dans l'autre colonne.

Réduction	
appel par nom	appel par valeur
(μ_n)	$(\tilde{\mu}_v)$
(μ)	$(\tilde{\mu})$
Syntaxe	
terme	contexte d'évaluation
valeur	contexte d'évaluation linéaire
x	α
$\mu\alpha.c$	$\tilde{\mu}x.c$

2.10 Plongement dans le λ-calcul par passage de continuation

Les deux restrictions CBN et CBV du sous-système $\mu\tilde{\mu}$ se plongent dans le λ-calcul via une traduction par passage de continuation (« continuation-passing style » ou CPS). Pour cela, on prend comme variables du λ-calcul la réunion disjointe des variables de terme (notées x, y, ...) et des variables de contexte (notées k_α, k_β, ...) du sous-système $\mu\tilde{\mu}$. Le plongement est paramétrique vis à vis des constructeurs d'expressions linéaires.

La traduction CPS par nom ^{n} utilise une traduction auxiliaire ^{nl} qui s'applique aux contextes d'évaluation linéaires. Les traductions sont définies par

$$
\begin{array}{lcllcl}
\alpha^{nl} & \triangleq & k_\alpha & x^n & \triangleq & x \\
(E_c)^{nl} & \triangleq & \ldots & (V_c)^n & \triangleq & \ldots \\
E^n & \triangleq & \lambda x.x\,E^{nl} & & & \\
(\tilde{\mu}x.c)^n & \triangleq & \lambda x.c^n & (\mu\alpha.c)^n & \triangleq & \lambda k_\alpha.c^n
\end{array}
$$
$$\langle v \| e\rangle^n \triangleq e^n\, v^n$$

La traduction CPS par valeur est définie symétriquement par

$$
\begin{array}{lcllcl}
\alpha^{v} & \triangleq & k_\alpha & x^{vl} & \triangleq & x \\
(E_c)^{v} & \triangleq & \ldots & (V_c)^{vl} & \triangleq & \ldots \\
 & & & V^v & \triangleq & \lambda k.k\,V^{vl} \\
(\tilde{\mu}x.c)^v & \triangleq & \lambda x.c^v & (\mu\alpha.c)^v & \triangleq & \lambda k_\alpha.c^v
\end{array}
$$
$$\langle v \| e\rangle^v \triangleq v^v\, e^v$$

Notons d'ailleurs que sous réserve d'une dualité au niveau des constructeurs d'expressions linéaires, on a

$$
\begin{array}{lcl}
(c^n)^\circ & = & (c^\circ)^v \\
(c^v)^\circ & = & (c^\circ)^n
\end{array}
$$

Les traductions CPS simulent correctement les deux restrictions du sous-système $\mu\tilde{\mu}$.

Proposition 6 *Si $c \to_n c'$ alors $c^n \xrightarrow{*}_\beta c'^n$. Si $c \to_v c'$ alors $c^v \xrightarrow{*}_\beta c'^v$.*

Les traductions CPS ainsi définies sont limitées. On verra dans les sections consacrées aux connecteurs des extensions intéressantes de ces traductions.

2.11 Complétude et séparabilité

Dans le sous-système $\mu\tilde{\mu}$, on peut clairement identifier deux notions de complètude pour les systèmes de réduction. La complétude opérationnelle caractérise la capacité à produire des formes normales tandis que la complétude observationnelle caractérise la capacité du système de réduction à identifier toute paire d'expressions dont les comportements sont non discriminables dans tout contexte d'observation.

Complétude opérationnelle

Soit $\rightarrow$ un système de réduction pour le sous-système $\mu\tilde{\mu}$. On dit que $\rightarrow$ est **complet opérationnellement** si pour toute commande c non normale (c'est-à-dire distincte des formes $\langle x \| E \rangle$ et $\langle V \| \alpha \rangle$), il existe c' tel que $c \xrightarrow{h} c'$ (réduction de tête). On a :

Proposition 7 *Les réductions $\rightarrow_n$ est $\rightarrow_v$ sont complètes opérationnellement dans toute extension du sous-système $\mu\tilde{\mu}$ sous réserve que toute coupure $\langle V_c \| E_c \rangle$ soit réductible.*

Complétude observationnelle

La structure du système $\mu\tilde{\mu}$ (tout comme d'ailleurs la prise en compte d'opérateurs de contrôle[1]) suggère que les contextes d'observation soient des commandes à un trou[2].

Si v est un terme, un **contexte d'observation pour l'appel par nom** (resp. **un contexte d'observation pour l'appel par valeur**) pour v est la donnée d'une substitution ρ CBN (resp. CBV) des variables libres de v et d'un contexte d'évaluation e.

Symétriquement, si e est un contexte d'évaluation, un **contexte d'observation pour l'appel par nom** (resp. **contexte d'observation pour l'appel par valeur**) pour e est la donnée d'une substitution ρ CBN (resp. CBV) des variables libres de e et d'un terme v.

Enfin, un **contexte d'observation pour l'appel par nom** (resp. **contexte d'observation pour l'appel par valeur**) pour une commande c est la donnée d'une substitution ρ CBN (resp. CBV) des variables libres de c.

Séparabilité pour les expressions normalisables

Les termes normalisables v_1 et v_2 sont **séparables en appel par nom** si, pour toutes commandes c_1 et c_2, il existe un contexte d'observation (ρ, e) pour l'appel par nom telle que $\langle v_1[\rho] \| e \rangle =_n c_1$ et $\langle v_2[\rho] \| e \rangle =_n c_2$. On dit qu'une égalité (générique) $v_1 = v_2$ entre termes normalisables fait **partie de la clôture observationnelle de l'appel par nom** si v_1 et v_2 ne sont pas séparables en appel par nom. Pareillement pour l'appel par valeur.

Notons que ces définitions ne reposent que sur les réductions $\rightarrow_n$ et $\rightarrow_v$ sans avoir à faire référence à la clôture observationnelle de ces réductions (en particulier sans avoir à mentionner les règles η considérées en section 2.12).

Dans la pratique, il n'est pas nécessaire de considérer la séparation pour n'importe quelles commandes. Pour l'appel par nom, il suffit de considérer deux commandes distinctes de la forme $\langle x_1 \| \alpha \rangle$ et $\langle x_2 \| \alpha \rangle$, puisque par substitution de $\mu\_.c_1$ à x_1 et de $\mu\_.c_2$ à x_2, on récupère la séparation vis à vis de toutes commandes. De même, pour l'appel par valeur, il suffit de considérer deux commandes distinctes de la forme $\langle x \| \alpha_1 \rangle$ et $\langle x \| \alpha_2 \rangle$. Ainsi, on retombe pour l'appel par nom sur la définition standard de la séparation utilisée dans le théorème de Böhm (cf par exemple l'introduction de l'article de synthèse de Dezani et Giovannetti [DG01]).

De la même manière, on obtient que deux termes (normalisables) v_1 et v_2 font partie de la clôture observationnelle de l'appel par nom ssi, à chaque fois que $\langle v_1[\rho] \| e \rangle =_n \langle x_1 \| \alpha \rangle$ et $\langle v_2[\rho] \| e \rangle =_n \langle x_2 \| \alpha \rangle$ pour e et ρ CBN, alors $x_1 = x_2$. Et similairement pour l'appel par valeur.

Deux contextes d'évaluation normalisables e_1 et e_2 sont **séparables en appel par nom** si, pour toutes commandes c_1 et c_2, il existe un contexte d'observation (ρ, v) pour l'appel par nom telle que $\langle v \| e_1[\rho] \rangle =_n c_1$ et $\langle v \| e_2[\rho] \rangle =_n c_2$. On dit qu'une égalité (générique) $e_1 = e_2$ entre contextes d'évaluation normalisables fait **partie de la clôture observationnelle de l'appel par nom** si e_1 et e_2 ne sont pas séparables en appel par nom. Et pareillement pour l'appel par valeur.

---

[1] Dans le calcul $\lambda_{\mathcal{C}}$ de Felleisen et Hieb [FFKD86, FH92], les différentes expressions $\mathcal{C}(\lambda k.kt)$, $\mathcal{C}(\lambda k.t)$ et $t[k \leftarrow \lambda x.\mathcal{A}(x)]$ ont la même sémantique (par exemple, leur exécution dans une machine à environnement dont la continuation est $\mathtt{tp}$ produiront $t[k \leftarrow \lambda x.\mathcal{C}(\lambda\_.\mathtt{tp}\, x)]$ dans les trois cas ; de même, leur sémantique par passage de continuation donne $t[k \leftarrow \lambda x.x]$ dans les trois cas). Cela poussa Sabry et Felleisen [SF93] à ne considérer que les termes de la forme $\mathcal{A}(t)$ (cf leur lemme 19). L'analyse de Ariola *et al* [AH03, AHS05a] montre que $\mathcal{A}(t)$ est représentable sous la forme $\mathcal{C}(\lambda\_.\mathtt{tp}\, t)$ et que seule la partie $\mathtt{tp}$ de $\mathcal{A}$ est nécessaire pour exprimer le lemme 19 de Sabry et Felleisen. Dans un calcul comme le calcul $\lambda\mu$ de Parigot (ou son équivalent basé sur $\mathcal{C}$ de Ariola *et al* [AH03, AHS05a]), cela correspond à ne considérer comme observations que les termes de la forme $[\alpha]t$ (cf aussi [AFH05]).

[2] Cf aussi la définition de la réalisabilité classique basée sur un ensemble d'observations de type $\perp$ par Krivine [Kri04].

Deux commandes normalisables c_1 et c_2 sont **séparables en appel par nom** si pour tous c_1' et c_2', il existe un contexte d'observation ρ pour l'appel par nom telle que $c_1[\rho] =_n c_1'$ et $c_2[\rho] =_n c_2'$. On dit qu'une égalité (générique) $c_1 = c_2$ entre commandes normalisables fait **partie de la clôture observationnelle de l'appel par nom** si c_1 et c_2 ne sont pas séparables en appel par nom. Et pareillement pour l'appel par valeur.

Comme pour la séparation des termes, il n'est pas nécessaire de considérer la séparation vers n'importe quelles commandes. Pour l'appel par nom, il suffit de considérer deux commandes distinctes de la forme $\langle x_1 \| \alpha\rangle$ et $\langle x_2 \| \alpha\rangle$ et pour l'appel par valeur, il suffit de considérer deux commandes distinctes de la forme $\langle x \| \alpha_1\rangle$ et $\langle x \| \alpha_2\rangle$.

Séparabilité en présence d'expressions non normalisables

Autant le système $\mu\tilde{\mu}$ pur est fortement normalisable, autant ses extensions ne le sont pas forcément. Il est clair alors que pour pouvoir séparer une expression normalisable d'une expression non normalisable, il faut pouvoir observer la non-terminaison. On généralise alors la définition de la séparabilité de telle sorte qu'elle s'applique, que le calcul soit constitué uniquement d'expressions normalisables ou non. Les termes v_1 et v_2 sont **séparables en appel par nom** s'il existe un contexte d'observation (ρ, e) pour l'appel par nom telle que l'une des conditions suivantes est vérifiée :

- pour tous c_1 et c_2, $\langle v_1[\rho] \| e\rangle =_n c_1$ et $\langle v_2[\rho] \| e\rangle =_n c_2$,
- pour un certain c_1 normal, $\langle v_1[\rho] \| e\rangle =_n c_1$ et $\langle v_2[\rho] \| e\rangle$ n'est pas normalisable pour $\rightarrow_n$,
- pour un certain c_2 normal, $\langle v_2[\rho] \| e\rangle =_n c_2$ et $\langle v_1[\rho] \| e\rangle$ n'est pas normalisable pour $\rightarrow_n$.

On dit alors qu'une égalité (générique) $v_1 = v_2$ entre termes fait **partie de la clôture observationnelle de l'appel par nom** si v_1 et v_2 ne sont pas séparables en appel par nom. Et pareillement pour l'appel par valeur. Notons que si le calcul contient de manière certaines des commandes non normalisables, alors la première clause de la définition de la séparabilité est redondante (il suffit de prendre un c_2 non normalisable).

Comme dans le cas de la séparabilité de termes normalisables, on peut se restreindre dans la première clause de la définition à prendre les commandes $\langle x_1 \| \alpha\rangle$ et $\langle x_2 \| \alpha\rangle$ pour x_1 et x_2 distincts. Ainsi, que $v_1 = v_2$ fasse partie de la clôture observationnelle de l'appel par nom signifie que pout tout (ρ, e) pour l'appel par nom, les deux conditions suivantes sont satisfaites :

- pour tout x_1 et x_2, $\langle v_1[\rho] \| e\rangle =_n \langle x_1 \| \alpha\rangle$ et $\langle v_2[\rho] \| e\rangle =_n \langle x_2 \| \alpha\rangle$ implique $x_1 = x_2$,
- il existe c_1 normal tel que $\langle v_1[\rho] \| e\rangle =_n c_1$ ssi il existe c_2 normal tel que $\langle v_2[\rho] \| e\rangle =_n c_2$,

où, là aussi, la première clause est redondante si on a accès à des observations non normalisables. En particulier, en présence d'observations non normalisables, on retombe sur la définition habituelle de l'équivalence observationnelle (cf encore, et par exemple, l'article de synthèse de Dezani et Giovannetti [DG01]).

Concluons cette discussion en montrant que si v_1 et v_2 sont des termes normaux pour lesquels on a montré l'appartenance de l'égalité $v_1 = v_2$ à la clôture observationnelle tant de l'appel par nom que de l'appel par valeur, alors toute instance, même non normalisable, de l'égalité $v_1 = v_2$ fait partie de la clôture observationnelle de l'appel par nom et de l'appel par valeur. Ceci est dû à la stabilité de la séparabilité CBN par substitution CBN et de la séparabilité CBV par substitution CBV.

Comparaison avec le λ-calcul

Le λ-calcul standard, c'est-à-dire le λ-calcul en appel par nom, tient d'une sorte de miracle. Trois constructeurs, une règle de réduction suffisante pour obtenir des formes normales de tête (c'est-à-dire la complétude opérationnelle), une règle de conversion supplémentaire pour obtenir la complétude observationnelle : tout est en place pour une théorie du calcul admise comme un modèle de la réalité du calculable.

En comparaison, le λ-calcul en appel par valeur paraît beaucoup plus complexe. La règle de réduction β_v est suffisante pour obtenir des formes normales de tête à partir d'expressions closes non divergentes, mais sa clôture observationnelle est plus complexe que celle du λ-calcul en appel par nom. En particulier, elle inclut la clôture observationnelle de sa traduction CPS. Sabry et Felleisen [SF93] ont décrit une axiomatique du λ-calcul CBV complète vis à vis de la CPS (Moggi [Mog88] a une axiomatique équivalente complète vis à vis d'une sémantique monadique). Cette axiomatique contient

$$\begin{array}{llcll}
(let_{lift}) & E[(\lambda y.t)\,u] & = & (\lambda y.E[t])\,u & \\
(\eta_{let}) & (\lambda x.E[x])\ t & = & E[t] & \text{si } x \text{ n'apparaît pas dans } E \\
(\eta_v) & \lambda x.(y\,x) & = & y &
\end{array}$$

où E est défini par $E ::= \Box \mid E\ M \mid (\lambda x.M)\,E$.

On peut connecter le λ-calcul par valeur et le calcul $\mu\tilde{\mu}_v$ étendu avec l'abstraction et l'application (voir la section 3.13.4). En fait, les deux premières règles s'interprètent même dans le sous-système $\mu\tilde{\mu}_v$ pur à partir du moment où l'on interprète un β-rédex comme un `let-in`. Ainsi $E[(\lambda y.t)\,u]$ et $(\lambda y.E[t])\,u$ peuvent s'interpréter comme $\langle \mu\alpha.\overline{\langle u \| \tilde{\mu} y.\overline{\langle t \| \alpha\rangle}\rangle} \| E\rangle$ et $\langle u \| \tilde{\mu} x.\overline{\langle t \| E\rangle}\rangle$ et $(\lambda x.E[x])\ t$ et $E[t]$ peuvent s'interpréter comme $\langle t \| \tilde{\mu} x.\overline{\langle x \| E\rangle}\rangle$ et $\langle t \| E\rangle$.

Ce que nous apprend donc le calcul symétrique, c'est que la première règle est une règle de calcul : elle permet de mettre potentiellement en évidence un rédex $E[t]$ (par exemple si E est de la forme $(\lambda x.u)\,\Box$ et t est une valeur). Autrement dit, c'est une coupure commutative (du même style que la commutation de la règle d'élimination de la disjonction avec les autres règles d'élimination en déduction naturelle – cf Prawitz [Pra65]).

En revanche, ce que continue de nous apprendre le calcul symétrique, c'est que la règle η_{let} est de même nature que de la règle $\eta_{\tilde{\mu}}$.

Ceci pose la question de trouver un critère permettant de distinguer, dans les axiomatiques de Sabry-Felleisen et Moggi, le sous-ensemble des règles qui sont calculatoires (qui, informellement, assurent la propriété de la sous-formule) et le sous-ensemble des règles (dont les règles d'η-conversion) dont le seul rôle est la stabilité observationnelle.

Ainsi le statut de la complétude opérationnelle est incertain dans λ-calcul en appel par valeur. Ceci est dû au fait que le rédex dit de tête d'une expression ne se trouve pas nécessairement à la racine de l'expression. L'analyse de la correspondance avec le système $\mu\tilde{\mu}$ étendu avec abstraction et contexte applicatif suggère de distinguer entre ce que nous appelerons complétude opérationnelle faible et complétude opérationnelle forte.

Par complétude opérationnelle faible, on qualifie la capacité du système de réduction à produire à partir d'expressions closes des formes « construites », c'est-à-dire des expressions commençant par un constructeur. Ainsi, la β_v-réduction est suffisante à la complétude opérationnelle faible.

Par complétude opérationnelle forte, on qualifie la capacité du système de réduction à révéler des rédex résiduels dans des expressions non nécessairement closes. Typiquement β_v n'est pas complet en ce sens. Par exemple, $(\lambda x.\lambda z.x)(y_1 y_2)y$ est en forme normale vis à vis de β_v mais, en appliquant la règle let_{lift}, on peut poursuivre la réduction vers $(\lambda x.((\lambda z.x)y))(y_1 y_2)$ puis $(\lambda x.x)(y_1 y_2)$, voire encore vers $y_1 y_2$.

2.12 Complétude observationnelle et η-conversion

Aux systèmes de réduction $\mu_n\tilde{\mu}$ et $\mu\tilde{\mu}_v$, il convient d'ajouter deux règles d'η-conversion qui identifient des expressions observationnellement égales. Ces règles, valables tant pour la restriction à l'appel par nom que pour la restriction à l'appel par valeur sont les suivantes :

$$\begin{array}{llcll}
(\eta_\mu) & \mu\alpha.\langle V \| \alpha\rangle & = & V & \alpha \text{ non libre dans } V \\
(\eta_{\tilde{\mu}}) & \tilde{\mu} x.\langle x \| E\rangle & = & E & x \text{ non libre dans } E
\end{array}$$

Remarquons au passage qu'il n'est nécessaire que de considérer des expressions linéaires V ou E puisque si les expressions n'étaient pas linéaires, les réductions μ ou $\tilde{\mu}$ s'appliqueraient.

Ces règles d'η-conversion pour μ et $\tilde{\mu}$ se plongent via la CPS en la règle standard d'η-conversion du λ-calcul. On a

Proposition 8 *Si $e =_{\eta_{\tilde{\mu}}} e'$ alors $e^n =_\eta e'^n$ et $e^v =_{\eta\beta} e'^v$. Si $v =_{\eta_\mu} v'$ alors $v^n =_{\eta\beta} v'^n$ et $v^v =_\eta v'^v$.*

Proposition 9 *$(\eta_{\tilde{\mu}})$ et (η_μ) font partie de la clôture observationnelle des restrictions en appel par nom et en appel par valeur*

PREUVE: Soit $e =_{\eta_{\tilde{\mu}}} e'$. Si e et e' étaient séparables pour la réduction CBN, alors, pour c et c' arbitraires, il existerait une observation CBN (v, σ) telle que $\langle v \| e[\sigma]\rangle =_n c$ et $\langle v \| e[\sigma]\rangle =_n c'$. Par traduction CPS CBN et proposition 6, on aurait alors $e^n[\sigma^n]\, v^n =_{\beta\eta} c^n$ et $e'^n[\sigma^n]\, v^n =_{\beta\eta} c'^n$ et e^n et e'^n seraient séparables dans le λ-calcul usuel. Or, par la proposition 8, on a $e^n =_\eta e'^n$ ce qui contredit leur séparabilité. Par conséquent, e et e' ne sont pas séparables. Le raisonnement est alors identique pour la séparation CBV et dual pour (η_μ). ∎

Dans le cas du sous-système $\mu\tilde{\mu}$ sans connecteurs (c'est-à-dire E_c et V_c vides), on a un résultat de complétude des réductions CBN et CBV étendus avec (η_μ) et $(\eta_{\tilde{\mu}})$, puisque les seules expressions normales sont $\langle x \| \alpha \rangle$ pour les commandes, x et $\mu\alpha.\langle x \| \beta \rangle$ pour les termes et α et $\tilde{\mu}x.\langle y \| \alpha \rangle$ pour les contextes d'évaluation linéaires.

Toutefois, il est probable, en appliquant le résultat de non séparabilité pour le $\lambda\mu$-calcul de David et Py [DP01], qu'on puisse montrer que les restrictions appel par nom et appel par valeur ne sont pas complètes en présence d'autres connecteurs vis à vis de l'équivalence observationnelle.

Intuitivement, cela s'explique au travers de la traduction CPS. Si c et c' sont des formes normales, on a $c =_{\mu\tilde{\mu}_v\eta_\mu\eta_{\tilde{\mu}}} c'$ implique $c^n =_{\beta\eta} c'^n$ ssi c^n et c'^n sont séparables dans le λ-calcul. Or il se peut que la séparation exige des contextes qui ne sont pas dans l'image de la traduction CPS, des contextes donc, qui n'ont pas de correspondant dans le calcul $\overline{\lambda}\mu\tilde{\mu}$.

On étudiera dans le chapitre 4 une extension du calcul $\overline{\lambda}\mu\tilde{\mu}$ avec délimiteurs de continuations. Ces délimiteurs de continuation ont la propriété de compléter la sémantique des calculs avec opérateurs de contrôle (cf Felleisen-Sitaram [SF90] et Filinski [Fil94]). On conjecture qu'il en est de même dans le $\overline{\lambda}\mu\tilde{\mu}$.

2.13 Fragment intuitionniste

On définit le fragment intuitionniste minimal du sous-système $\mu\tilde{\mu}$ en imposant que chaque occurrence de μ lie une et une seule variable de contexte d'évaluation.

On observe alors par induction que tout contexte d'évaluation et toute commande ont nécessairement une et une seule variable de contexte d'évaluation libre et que tout terme est clos vis à vis des variables de contexte d'évaluation. De plus les liaisons par μ ne se chevauchent pas si bien que dans le cas du système $\mu\tilde{\mu}$ pur, l'ensemble des variables de contexte d'évaluation peut-être réduit à un singleton. Notons alors $\star$ pour l'unique nom de variable nécessaire.

Le sous-système $\mu\tilde{\mu}$ intuitionniste est défini par la syntaxe

$$\begin{array}{lcl} c & ::= & \langle v \| e \rangle \\ v & ::= & x \mid \mu\star.c \\ e & ::= & \star \mid \tilde{\mu}x.c \end{array}$$

La linéarité de la liaison de μ a pour conséquence que la règle (μ) commute avec la règle $(\tilde{\mu})$ sans jamais ni l'effacer ni la dupliquer. En particulier, on a :

Proposition 10 *Si $c \rightarrow_\mu c_1$ et $c \rightarrow_{\tilde{\mu}} c_2$, alors il existe c' tel que $c_1 \rightarrow_{\tilde{\mu}} c'$ et $c_2 \xrightarrow{*}_\mu c'$.*

Preuve: Si les deux réductions sont appliquées dans des sous-expressions disjointes, elles commutent simplement. Sinon, soit $c_1 \triangleq \langle v \| \tilde{\mu}x.c \rangle$ la sous-commande sujette à la réduction $(\tilde{\mu})$ et soit $c_2 \triangleq \langle \mu\alpha.c' \| e \rangle$ la sous-commande sujette à la réduction (μ). Si c_2 est sous-commande de c ou si c_1 est sous-commande de c', alors on a une commutation simple par compatibilité de la réduction avec la substitution. Si c_2 est sous-commande de v alors la réduction $(\tilde{\mu})$ peut effacer ou dupliquer c_2. Le diagramme se clôt en appliquant un nombre arbitraire de fois (μ). Enfin, si c_1 est sous-commande de e, il continuera d'apparaître sous forme unique après réduction (μ) et une unique étape $(\tilde{\mu})$ est nécessaire pour faire converger la réduction. ∎

Une autre conséquence est que l'appel par valeur devient une restriction de l'appel par nom.

Proposition 11 *Si $c \xrightarrow{*}_v \langle v \| \star \rangle$, alors $c \xrightarrow{*}_n \langle v \| \star \rangle$.*

En particulier, le système $\mu\tilde{\mu}$ dit « non-déterministe » est déterministe dans sa restriction intuitionniste.

2.14 Modélisation des machines abstraites

Du fait que le rédex de tête se trouve à la racine du terme, le sous-système $\mu\tilde{\mu}$ et plus généralement le calcul $\overline{\lambda}\mu\tilde{\mu}$ se prête bien à la modélisation des machines abstraites. À cette fin, on enrichit le calcul avec des environnements et des constantes.

$$\begin{array}{lcl} [\rho] & ::= & [\,] \mid [x := V; \rho] \mid [\alpha := e; \rho] \\ v & ::= & \ldots \mid v[\rho] \\ e & ::= & \ldots \mid e[\rho] \end{array}$$

Voici par exemple le cœur d'une machine abstraite pour l'évaluation en appel par valeur.

Contrôle donné à l'évaluation

$$\begin{array}{lcl} \langle \mu\alpha.\langle\overline{v\|e'}\rangle[\rho]\|e\rangle & \rightarrow & \langle v[\alpha := e; \rho]\|e'[\alpha := e; \rho]\rangle \\ \langle x[\rho]\|e\rangle & \rightarrow & \langle \rho(x)\|e\rangle \end{array}$$

Contrôle donné au contexte d'évaluation

$$\begin{array}{lcl} \langle V_c[\rho']\|\tilde{\mu}x.\langle\overline{v\|e}\rangle[\rho]\rangle & \rightarrow & \langle v[x := V_c[\rho']; \rho]\|e[x := V_c[\rho']; \rho]\rangle \\ \langle V_c[\rho']\|\alpha[\rho]\rangle & \rightarrow & \langle V_c[\rho']\|\rho(\alpha)\rangle \end{array}$$

2.15 Typage simple du sous-système $\mu\tilde{\mu}$ non-déterministe

On donne les règles de typage simple du sous-système $\mu\tilde{\mu}$ non-déterministe et en particulier, on se donne un ensemble de formules dont les éléments sont dénotés par les lettres A, B, etc. Les règles de typage sont celles déjà mentionnées en section 1.2 à ceci près que les contextes de typage, dénotés par les lettres Γ ou Δ, sont maintenant des ensembles de formules indexées par des variables. Comme son nom le suggère, un contexte d'évaluation linéaire est un contexte d'évaluation qui ne provient pas d'une contraction. Pour expliciter la distinction entre expressions linéaires et non linéaires, nous introduisons cinq types de séquents.

valeurs	$\Gamma \vdash V : A\,;\,\Delta$
termes	$\Gamma \vdash v : A \mid \Delta$
contextes d'évaluation linéaires	$\Gamma\,;\,E : A \vdash \Delta$
contextes d'évaluation	$\Gamma \mid e : A \vdash \Delta$
commandes	$c : (\Gamma \vdash \Delta)$

Le passage d'une expression linéaire à une expression dont on oublie l'information de linéarité est réalisé par des règles de « déréliction ». Le système de typage $LK_{\mu\tilde{\mu}}$ obtenu est le suivant :

Typage simple du système non-déterministe

$$Ax_L\ \frac{}{\Gamma;\Gamma \mid \alpha : A \vdash \alpha : A, \Delta} \qquad Ax_R\ \frac{}{\Gamma, x : A \vdash x : A \mid \Delta}$$

$$\tilde{\mu}\ \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta} \qquad \mu\ \frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}$$

$$Der_L\ \frac{\Gamma\,;\,E_c : A \vdash \Delta}{\Gamma \mid E_c : A \vdash \Delta} \qquad Der_R\ \frac{\Gamma \vdash V_c : A\,;\,\Delta}{\Gamma \vdash V_c : A \mid \Delta}$$

$$Cut\ \frac{\Gamma \vdash v : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{c : (\Gamma \vdash \Delta)}$$

Précisons que toute expression ne faisant pas référence aux constructeurs d'expressions linéaires est typable : il suffit d'attribuer le même type à toutes les variables. De même la préservation du typage par les règles de réduction (μ) et $(\tilde{\mu})$ est sans surprise.

Proposition 12 *Les règles de réduction non-déterministes du sous-système $\mu\tilde{\mu}$ préservent le typage. Il en est de même des règles de conversion η.*

2.16 Typage simple du sous-système $\mu\tilde{\mu}$ en appel par nom

Le système de typage $LK_{\mu_n\tilde{\mu}}$ du sous-système $\mu\tilde{\mu}$ en appel par nom introduit l'invariant supplémentaire que les variables de contexte sont linéaires. On obtient donc les règles d'inférence suivantes pour lesquelles les contextes linéaires et les constructeurs de contextes linéaires sont typés par le même type de séquents.

Typage simple de l'appel par nom

$$Ax_L^n \ \frac{}{\Gamma\,;\,\alpha : A \vdash \alpha : A, \Delta} \qquad Ax_R \ \frac{}{\Gamma, x : A \vdash x : A \,|\, \Delta}$$

$$\tilde{\mu} \ \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}x.c : A \vdash \Delta} \qquad \mu \ \frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \,|\, \Delta}$$

$$Der_L \ \frac{\Gamma\,;\,E : A \vdash \Delta}{\Gamma \,|\, E : A \vdash \Delta} \qquad Der_R \ \frac{\Gamma \vdash V_c : A\,;\,\Delta}{\Gamma \vdash V_c : A \,|\, \Delta}$$

$$Cut \ \frac{\Gamma \vdash v : A \,|\, \Delta \quad \Gamma \,|\, e : A \vdash \Delta}{\langle v \| e \rangle : (\Gamma \vdash \Delta)}$$

Proposition 13 *Les règles de réduction en appel par nom du sous-système $\mu\tilde{\mu}$ préservent le typage en appel par nom. Il en est de même des règles de conversion η.*

2.17 Typage simple du sous-système $\mu\tilde{\mu}$ en appel par valeur

Le système de typage $LK_{\mu\tilde{\mu}_v}$ du sous-système $\mu\tilde{\mu}$ en appel par valeur est dual. Il introduit l'invariant que les variables de terme sont linéaires. On obtient les règles d'inférence suivantes pour lesquelles les valeurs et les constructeurs de valeurs sont typés par le même type de séquents.

Typage simple de l'appel par valeur

$$Ax_L \ \frac{}{\Gamma \,|\, \alpha : A \vdash \alpha : A, \Delta} \qquad Ax_R^v \ \frac{}{\Gamma, x : A \vdash x : A\,;\,\Delta}$$

$$\tilde{\mu} \ \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}x.c : A \vdash \Delta} \qquad \mu \ \frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \,|\, \Delta}$$

$$Der_L \ \frac{\Gamma\,;\,E_c : A \vdash \Delta}{\Gamma \,|\, E_c : A \vdash \Delta} \qquad Der_R \ \frac{\Gamma \vdash V : A\,;\,\Delta}{\Gamma \vdash V : A \,|\, \Delta}$$

$$Cut \ \frac{\Gamma \vdash v : A \,|\, \Delta \quad \Gamma \,|\, e : A \vdash \Delta}{\langle v \| e \rangle : (\Gamma \vdash \Delta)}$$

Proposition 14 *Les règles de réduction en appel par valeur du sous-système $\mu\tilde{\mu}$ préservent le typage en appel par valeur. Il en est de même des règles de conversion η.*

2.18 $\neg\neg$-traduction

Au niveau du typage, la cible des traductions CPS est un λ-calcul typé dans lequel on distingue un type particulier que l'on note $\bot_0$. Dans ce λ-calcul, on note $\neg A \triangleq A \to \bot_0$. Les traductions CPS de termes simplement typés donnent des termes typés de type une $\neg\neg$-traduction des termes originaux. Comme le système de typage n'a pas de connecteurs, la traduction des types est assez sommaire.

Proposition 15

On a, pour l'appel par nom
- $\Gamma \vdash v : A \,|\, \Delta$ *implique* $\neg\Gamma, \Delta \vdash v^v : \neg A$
- $\Gamma\,;\, E : A \vdash \Delta$ *implique* $\neg\Gamma, \Delta \vdash E^{ln} : A$
- $\Gamma \,|\, e : A \vdash \Delta$ *implique* $\neg\Gamma, \Delta \vdash e^n : \neg\neg A$
- $(c : \Gamma \vdash \Delta)$ *implique* $\neg\Gamma, \Delta \vdash c^n : \bot$

On a, pour l'appel par valeur
- $\Gamma \vdash V : A\,;\, \Delta$ *implique* $\Gamma, \neg\Delta \vdash V^{lv} : A$
- $\Gamma \vdash v : A \,|\, \Delta$ *implique* $\Gamma, \neg\Delta \vdash v^v : \neg\neg A$
- $\Gamma \,|\, e : A \vdash \Delta$ *implique* $\Gamma, \neg\Delta \vdash e^v : \neg A$
- $(c : \Gamma \vdash \Delta)$ *implique* $\Gamma, \neg\Delta \vdash c^v : \bot$

Notons que dans le cas de l'appel par nom, la traduction sur les types n'interprète pas A d'une manière logiquement équivalente de part et d'autre de l'implication. Au contraire, la valeur logique de A est niée par la traduction appel par nom. Ceci n'a pas d'importance puisqu'ici les formules ont un rôle inerte et se comportent comme des variables atomiques. On pourra voir en section 3.2.10 une extension de la traduction au cas de l'implication.

On verra dans la section suivante que l'extension du typage avec types intersection et union n'est pas aussi directe.

2.19 Types intersection et union

On étend le typage avec des types intersection et union finis[3]. On étend l'ensemble des formule avec deux nouveaux constructeurs de type.

$$A \quad ::= \quad \ldots \,|\, A \cap B \,|\, A \cup B$$

L'ajout sans restriction de type union au sous-système appel par nom et de type intersection au sous-système appel par valeur pose des problèmes de typage de la réduction.

Le problème s'exprime ainsi : si $\tilde{\mu}x.c$ est de type $A \cup B$ parce qu'à la fois de type A (avec x de type A) et de type B (avec x de type B) tandis que $\mu\alpha.c'$ est de type $A \cup B$ avec α de type $A \cup B$, alors la réduction de $\langle \mu\alpha.c' \| \tilde{\mu}x.c \rangle$ en appel par nom produit une expression $c[x \leftarrow \mu\alpha.c']$ qui n'est a priori pas typable. Pour le typer, il faudrait pouvoir contraindre $\mu\alpha.c'$ à être soit de type A soit de type B, ce qui nécessite que α soit de type A ou de type B, ce qui nécessite que tous les termes en interaction avec α dans c' (quitte à appliquer une expansion $\eta_{\tilde{\mu}}$ pour que tout α soit en interaction) puissent eux-mêmes se voir attribuer le type A ou B. Si l'un de ces termes est une variable libre de type $A \cup B$, son type est contraint par le contexte de typage et il ne peut être changé. Inversement, on pourrait essayer de contraindre x à être de type $A \cup B$ dans c, ce qui nécessite que tout sous-terme e de type A en interaction avec x (quitte à appliquer une expansion η_μ pour que tout x soit en interaction) puisse être contraint à être de type $A \cup B$. Comme ce sous-terme est aussi de type B dans le typage de $\tilde{\mu}x.c$ de type B, on pourrait penser pouvoir unifier les deux dérivations de telle sorte que e ait le type $A \cup B$. Cependant, cela ne fonctionnera a priori pas en présence de connecteurs car le connecteur peut être de type union ou intersection sans que ses composantes le soient[4] (cf Pierce [Pie91] pour une formulation du problème en λ-calcul).

Une solution communément admise est de restreindre en appel par nom les contextes d'évaluation de type intersection à provenir soit de contextes d'évaluation linéaires, soit de la liaison d'une variable

[3] Plus généralement, l'extension s'applique aux intersections et unions infinies, ce qui permet en particulier de représenter les quantifications universelles et existentielles non calculatoires.

[4] Par exemple, en présence de l'implication (cf section 3.2), le sous-terme $\langle y \| x \cdot \beta \rangle$ avec y de type $(A \to C) \cap (B \to C)$ et β de type C est typable tant pour x de type A que pour x de type B sans pouvoir être typé avec x de type $A \cup B$.

déjà de type intersection. Symétriquement pour l'appel par valeur. Dans le cas du sous-système $\mu\tilde{\mu}$ non déterministe, cela signifie conjuguer les deux restrictions. Pour les versions appel par nom et appel par valeur du sous-système $\mu\tilde{\mu}$, on obtient les règles de typage suivantes pour l'union et l'intersection :

Appel par nom

$$\cap^n_{L_i}\ \frac{\Gamma\,;\, E : A_i \vdash \Delta}{\Gamma\,;\, E : A_1 \cap A_2 \vdash \Delta} \qquad \cap_R\ \frac{\Gamma \vdash v : A_1 \,|\, \Delta \qquad \Gamma \vdash v : A_2 \,|\, \Delta}{\Gamma \vdash v : A_1 \cap A_2 \,|\, \Delta}$$

$$\cup_L\ \frac{\Gamma \,|\, e : A_1 \vdash \Delta \qquad \Gamma \,|\, e : A_2 \vdash \Delta}{\Gamma \,|\, e : A_1 \cup A_2 \vdash \Delta} \qquad \cup_{R_i}\ \frac{\Gamma \vdash v : A_i \,|\, \Delta}{\Gamma \vdash v : A_1 \cup A_2 \,|\, \Delta}$$

Appel par valeur

$$\cap_{L_i}\ \frac{\Gamma \,|\, e : A_i \vdash \Delta}{\Gamma \,|\, e : A_1 \cap A_2 \vdash \Delta} \qquad \cap_R\ \frac{\Gamma \vdash v : A_1 \,|\, \Delta \qquad \Gamma \vdash v : A_2 \,|\, \Delta}{\Gamma \vdash v : A_1 \cap A_2 \,|\, \Delta}$$

$$\cup_L\ \frac{\Gamma \,|\, e : A_1 \vdash \Delta \qquad \Gamma \,|\, e : A_2 \vdash \Delta}{\Gamma \,|\, e : A_1 \cup A_2 \vdash \Delta} \qquad \cup^v_{R_i}\ \frac{\Gamma \vdash V : A_i\,;\, \Delta}{\Gamma \vdash V : A_1 \cup A_2\,;\, \Delta}$$

Proposition 16 *La réduction préserve le typage.*

2.20 Normalisation par évaluation

La syntaxe du calcul des séquents étant plus régulière que celle de la déduction naturelle, et le rédex de tête étant placé en tête (et non enfoui sous une série d'applications comme en déduction naturelle), il est intéressant d'observer le contenu calculatoire des preuves de normalisation par réductibilité [Her01][5].

On donne ci-dessous le contenu calculatoire de la preuve de terminaison de la restriction à appel par nom. Pour cette preuve, les ensembles de réductibilité sont définis par induction sur la catégorie des expressions : contextes d'évaluation linéaire, termes, contexte d'évaluation, dans cet ordre.

On se donne en paramètre une définition de la réductibilité pour les constructeurs de contexte d'évaluation linéaire. On note ce paramètre $[\![|| A \vdash]\!]$ pour A donné. On note WN pour l'ensemble des expressions (faiblement) normalisables. Les ensembles de réductibilité sont définis par

- $E \in [\![\,;\, A \vdash]\!]$ ssi E est une variable ou $E \in [\![|| A \vdash]\!]$
- $v \in [\![\vdash A \,|\,]\!]$ ssi pour tout contexte d'évaluation linéaire $E \in [\![A \vdash]\!]$, $\langle v \| E\rangle$ est WN
- $e \in [\![A \vdash]\!]$ ssi pour tout terme $v \in [\![\vdash A \,|\,]\!]$, $\langle v \| e\rangle$ est WN

On note en particulier que tout contexte d'évaluation linéaire vérifie la dernière clause, si bien qu'on a $[\![\,;\, A \vdash]\!] \subset [\![A \vdash]\!]$ comme attendu.

On montre par induction sur la catégorie de l'expression (puis sur le type) que les ensembles de réductibilité sont entre les variables et les expressions WN et qu'ils sont stables par expansion de la réduction.

Lemme 1 *Soit A un type, on a :*
- $\mathcal{A} \subset [\![\,;\, A \vdash]\!] \subset [\![\,|\, A \vdash]\!] \subset WN$
- $\mathcal{X} \subset [\![\vdash A \,|\,]\!] \subset WN$
- *WN, $[\![\,;\, A \vdash]\!]$, $[\![\,|\, A \vdash]\!]$ et $[\![\vdash A \,|\,]\!]$ sont stables par expansion*

[5]Cette preuve qui traite de la réduction du calcul $\overline{\lambda}\mu\tilde{\mu}$ étendu avec un opérateur de substitution explicite est incomplète. En particulier, la preuve du lemme d'affaiblissement y est incorrecte. Toutefois, la structure de preuve, si l'on oublie le traitement des substitutions explicites (pour lequel la notion de réductibilité n'apporte rien), reste pertinente pour prouver la terminaison du calcul sans substitution explicite.

Soit τ une substitution CBN. On dit que $\tau \in [\![\Gamma \vdash \Delta]\!]$ si pour tout $x : A$ dans Γ, $\tau(x) \in [\![\vdash A\,|\,]\!]$ et pour tout $\alpha : A$ dans Δ, $\tau(\alpha) \in [\![\,;A \vdash]\!]$. On montre par induction sur la structure des expressions que les termes typables sont dans les ensembles de réductibilité :

Lemme 2 (adéquation) *Soit* $\tau \in [\![\Gamma \vdash \Delta]\!]$*, on a :*
- $c : (\Gamma \vdash \Delta)$ *implique* $c[\tau] \in WN$
- $\Gamma \vdash v : A\,|\,\Delta$ *implique* $v[\tau] \in [\![\vdash A\,|\,]\!]$
- $\Gamma\,|\,e : A \vdash \Delta$ *implique* $e[\tau] \in [\![\,|\,A \vdash]\!]$

On en déduit

Proposition 17
- *toute commande typée est dans WN*
- *tout terme typé est dans WN*
- *tout contexte d'évaluation typé est dans WN*

L'extraction par réalisabilité modifiée de ces lemmes donne un programme de normalisation (par exemple en définissant $c \in$ WN par $\exists c'$ normal tel que $c \to_n c'$). De manière directe, les commandes sont interprétées par des expressions dénotant des commandes en forme normale ; les contextes d'évaluation linéaires sont interprétés par des expressions les représentant ; les termes sont représentés par des fonctions qui produisent des commandes en forme normale quand appliquées à un contexte d'évaluation linéaire ; les contextes d'évaluation sont interprétés par des fonctions qui produisent des commandes en forme normale quand appliquées à des fonctions interprétant les termes.

```
(** Expressions *)

type command = term * context
 and value_cstr
 and term = VVar of string | Mu of string * command | ValCstr of value_cstr
 and linear_context_cstr
 and linear_context = EVar of string | LinCtxCstr of linear_context_cstr
 and context = MuTilde of string * command | LinCtx of linear_context

(** Ensembles de réductibilité *)

type command_red = term * context
 and linear_context_cstr_red
 and linear_context_red =
   EVarRed of string | LinCtxCstrRed of linear_context_cstr_red
 and term_red = linear_context_red -> command_red
 and context_red = term_red -> command_red

(** La réductibilité contient les variables et est contenue dans WN *)

(* A subset [| | |- |] *)
let embedvare a = (EVarRed a : linear_context_red)

(* [| | |- |] subset [| ; |- |] *)
let embed_linear_context (e:linear_context_red) = ((fun v -> v e):context_red)

(* [| | |- |] subset WN (pour extensions) *)
let rec embedsnlec e = failwith "To define"

(* [| | |- |] subset WN *)
and embedsnle = function
| EVarRed a -> EVar a
| LinCtxCstrRed e -> LinCtxCstr (embedsnlec e)
```

```
(* X subset [| |- ; |] *)
and embedvarv x = ((fun e -> (VVar x,LinCtx (embedsnle e))) : term_red)

(* [| |-; |] subset WN (pour extensions) *)
and embedsnv v (f:term_red) = match v with
| Mu (a,c) -> Mu (a,f (EVarRed a))
| v -> fst (f (EVarRed ""))

(* [| ;|- |] subset WN (pour extensions) *)
and embedsne e (g:context_red) = match e with
| MuTilde (x,c) -> MuTilde (x,g (embedvarv x))
| v -> snd (g (embedvarv ""))

(** Lemme d'adéquation *)

type term_env = (string * term_red) list
type context_env = (string * linear_context_red) list
type env = term_env * context_env

(* interpc : env -> command -> command_red *)
let rec interpc (rho:env) (v,e) = ((interpe rho e) (interpv rho v) : command)

(* interpv : env -> term -> term_red *)
and interpv (rho,sigma : env) = function
| VVar x -> List.assoc x rho
| Mu (a,c) -> fun (e:linear_context_red) -> interpc (rho,(a,e)::sigma) c
| ValCstr v -> fun (e:linear_context_red) -> interpvc (rho,sigma) (v,e)

(* interpvc : env -> value_cstr -> linear_context_red -> command_red *)
and interpvc (rho,sigma : env) (v,e) = failwith "To define"

(* interple : env -> linear_context -> linear_context_red *)
and interple (rho,sigma : env) = function
| EVar a -> List.assoc a sigma
| LinCtxCstr e -> LinCtxCstrRed (interplec (rho,sigma) e)

(* interpe : env -> context -> context_red *)
and interpe (rho,sigma : env) = function
| MuTilde (x,c) -> fun (v:term_red) -> interpc ((x,v)::rho,sigma) c
| LinCtx e -> embed_linear_context (interple (rho,sigma) e)

(* interplec : env -> linear_context_cstr -> linear_context_cstr_red *)
and interplec env = failwith "To define"

(** Calcul des variables libres *)

let rec remove x = function
| y::l when x=y -> l
| y::l -> y::remove x l
| [] -> []

let rec varsc (v,e) =
  let (vvx,vva) = varsv v in
  let (evx,eva) = varse e in
  (vvx@evx,vva@eva)
```

```
and varsv = function
| VVar x -> [x],[]
| Mu (a,c) -> let (vx,va) = varsc c in (vx,remove a va)
| ValCstr v -> failwith "To define"

and varsle = function
| EVar a -> [],[a]
| LinCtxCstr e -> failwith "To define"

and varse = function
| MuTilde (x,c) -> let (vx,va) = varsc c in (remove x vx,va)
| LinCtx e -> varsle e

(** Substitution à alpha-conversion près *)

let rec fresh x l = if List.mem_assoc x l then fresh (x^"'") l else x

let rec substc subst (v,e) = (substv subst v,subste subst e)

and substv (vs,es) = function
| VVar x -> List.assoc x vs
| Mu (a,c) -> let a' = fresh a es in Mu (a',substc (vs,(a,EVar a')::es) c)
| ValCstr v -> failwith "To define"

and substle (vs,es) = function
| EVar a -> List.assoc a es
| LinCtxCstr e -> failwith "To define"

and subste (vs,es) = function
| MuTilde (x,c) ->
  let x' = fresh x vs in MuTilde (x',substc ((x,VVar x')::vs,es) c)
| LinCtx e -> LinCtx (substle (vs,es) e)

(** Preuve de l'existence d'une forme normale *)

let varssubst (vx,va) =
  List.map (fun x -> (x,embedvarv x)) vx,
  List.map (fun a -> (a,embedvare a)) va

let normalc c = interpc (varssubst (varsc c)) c

let normalv v = embedsnv v (interpv (varssubst (varsv v)) v)

let normale e = embedsne e (interpe (varssubst (varse e)) e)
```

2.21 Appel par valeur paresseux et appel par nom paresseux

L'évaluation paresseuse (appelée aussi appel par nécessité) du λ-calcul est une optimisation de l'appel par nom qui partage l'évaluation des arguments qui sont effectivement nécessaires à la poursuite du calcul (nous faisons ici référence à Ariola-Felleisen [AF93])

D'un point de vue sémantique, appel par nom et appel par nécessité produisent les mêmes valeurs ou formes normales dans le λ-calcul pur (intuitionniste et avec non terminaison).

Quel sens prend l'évaluation paresseuse dans le contexte d'un système avec opérateurs de contrôle ? Que devient l'évaluation paresseuse sous le regard de la dualité ?

On peut déjà observer que l'évaluation paresseuse ne duplique pas les termes et ne substitue que des valeurs. En ce sens, c'est un appel par valeur. Appelons-le donc **appel par valeur paresseux**, ce qui laisse de la place pour le comportement dual que l'on pourra appeler **appel par nom paresseux**.

Contrairement à l'appel par valeur simple, l'appel par nécessité ne substitue une valeur que si elle est réellement nécessaire pour la suite du calcul. On peut donc formaliser l'évaluation paresseuse dans le sous-système $\mu\tilde{\mu}$ de la manière suivante :

Réduction en appel par valeur paresseux

$$\begin{array}{llll}
(\mu_n) & \langle\mu\alpha.c\|E\rangle & \xrightarrow{h}_{lv} & c[\alpha \leftarrow E] \\
(\tilde{\mu}_v) & \langle V\|\tilde{\mu}x.c\rangle & \xrightarrow{h}_{lv} & c[x \leftarrow V] \\
(\mu_{lv}) & \langle\mu\alpha.c\|\tilde{\mu}x.c'\rangle & \xrightarrow{h}_{lv} & c[\alpha \leftarrow \tilde{\mu}x.c'] \quad \text{si } x \text{ est } \textbf{nécessaire} \text{ dans } c' \\
(\tilde{\mu}_{lv}) & \langle\mu\alpha.c\|\tilde{\mu}x.c'\rangle & \xrightarrow{h}_{lv} & c' \quad\quad\quad\quad\quad\ \text{si } x \notin FV(c')
\end{array}$$

où la propriété x est nécessaire dans c est définie inductivement par

- x est nécessaire dans $\langle x\|E\rangle$,
- si x est nécessaire dans c' alors il est nécessaire dans $\langle\mu\alpha.c\|\tilde{\mu}y.c'\rangle$.

Il est à noter que la nouvelle contrainte est assez restrictive. En particulier, on perd la propriété que les commandes en forme normale de tête coïncident avec les expressions de la forme $\langle x\|E_c\rangle$ ou $\langle V_c\|\alpha\rangle$ ou $\langle x\|\alpha\rangle$. Par exemple, l'expression $\langle\mu\alpha.c\|\tilde{\mu}x.\langle\overline{y\|\alpha}\rangle\rangle$ est en forme normale (si c l'est) sans être de l'une des deux formes ci-dessus. Toutefois, les termes clos normaux sont de la forme

$$\langle\mu\alpha_1.c_1\|\tilde{\mu}x_1.\langle\overline{\mu\alpha_2.c_2\|\tilde{\mu}x_1.\dots.\langle\overline{V_c\|E_c}\rangle\dots}\rangle\rangle$$

pour $\langle V_c\|E_c\rangle$ normal.

Notre définition diffère légèrement de celle de Ariola et Felleisen [AF93] : une fois réduit en valeur un argument nécessaire est substitué partout où il est apparaît, alors que Ariola et Felleisen laissent explicites les substitutions individuelles de chaque variable. Cela ne change pas la sémantique.

Par dualité, on définira l'appel par nom paresseux de la sorte.

Réduction en appel par nom paresseux

$$\begin{array}{llll}
(\mu_n) & \langle\mu\alpha.c\|E\rangle & \xrightarrow{h}_{ln} & c[\alpha \leftarrow E] \\
(\tilde{\mu}_v) & \langle V\|\tilde{\mu}x.c\rangle & \xrightarrow{h}_{ln} & c[x \leftarrow V] \\
(\mu_{ln}) & \langle\mu\alpha.c\|\tilde{\mu}x.c'\rangle & \xrightarrow{h}_{lv} & c \quad\quad\quad\quad\quad\ \text{si } \alpha \notin FV(c) \\
(\tilde{\mu}_{ln}) & \langle\mu\alpha.c\|\tilde{\mu}x.c'\rangle & \xrightarrow{h}_{lv} & c'[x \leftarrow \mu\alpha.c] \quad \text{si } \alpha \text{ est nécessaire dans } c
\end{array}$$

Notons pour finir cet aperçu que l'appel par valeur paresseux, l'appel par nom paresseux, l'appel par nom et l'appel par valeur sont tous distincts entre eux (hors restriction intuitionniste bien sûr). En particulier, si on pose

$$c_0 \triangleq \langle\mu\alpha_1.\langle\overline{\mu\alpha_2.\langle\overline{V\|\alpha_2}\rangle\|\tilde{\mu}y.c}\rangle\|\tilde{\mu}x_1.\langle\overline{\mu\beta.c'\|\tilde{\mu}x_2.\langle\overline{x_2\|E}\rangle}\rangle\rangle$$

alors, on a

$$\begin{array}{lll}
c_0 & \rightarrow_n & c'[\beta \leftarrow E[x_2 \leftarrow \mu\beta.c']] \\
c_0 & \rightarrow_v & c[y \leftarrow V[\alpha_2 \leftarrow \tilde{\mu}y.c]] \\
c_0 & \rightarrow_{ln} & c[y \leftarrow \mu\alpha_2.\langle V\|\alpha_2\rangle] \\
c_0 & \rightarrow_{lv} & c'[\beta \leftarrow \tilde{\mu}x_2.\langle x_2\|E\rangle]\,,
\end{array}$$

la discrimination pouvant continuer dans un langage contenant des constructeurs d'expressions linéaires.

En présence de constructeurs d'expressions linéaires, on peut aussi considérer des règles de commutation permettant de retrouver des formes normales de la forme $\langle x\|E_c\rangle$, $\langle V_c\|\alpha\rangle$ ou $\langle x\|\alpha\rangle$. Par exemple, en présence d'abstraction (cf section 3.2), on peut ajouter à l'appel par valeur paresseux la règle :

$$\langle\mu\alpha.c\|\tilde{\mu}y.\langle\overline{\lambda x.v\|\beta}\rangle\rangle \rightarrow_{lv} \langle\lambda x.\mu\gamma.\langle\overline{\mu\alpha.c\|\tilde{\mu}y.\langle\overline{v\|\gamma}\rangle}\rangle\|\beta\rangle$$

2.22 Récursion et corécursion

La structure du sous-système $\mu\tilde{\mu}$ permet aussi une description duale de la récursion et de la corécursion.

$$\begin{array}{rcl} v & ::= & \ldots \mid \nu_x.v \\ e & ::= & \ldots \mid \tilde{\nu}_\alpha.e \end{array}$$

La corécursion s'obtient simplement par un opérateur de point fixe $\nu_x.v$ qui permet de définir un terme t faisant référence à lui-même. Par dualité, on attend de la récursion qu'elle corresponde à définir un contexte d'évaluation e pouvant faire référence à lui-même.

La notation μ souvent utilisée pour définir un point-fixe étant déjà utilisée, on utilise la notation $\tilde{\nu}$ pour un contexte d'évaluation récursif.

Les définitions récursives de termes et de contextes d'évaluation viennent avec les règles de réduction suivantes :

$$\begin{array}{lrcl} (\nu) & \nu_x.v & \stackrel{h}{\rightarrow} & v[x \leftarrow \nu_x.v] \\ (\tilde{\nu}) & \tilde{\nu}_\alpha.e & \stackrel{h}{\rightarrow} & e[\alpha \leftarrow \tilde{\nu}_\alpha.e] \end{array}$$

On verra en section 3.11 le lien entre fonctions récursives et contexte d'évaluation récursif.

Chapitre 3

Constructions logiques

3.1 Introduction

Maintenant que la structure calculatoire $\mu\tilde{\mu}$ est en place, nous étudions comment l'étendre avec un contenu logique et, pour cela, nous passons en revue diverses constructions associées à des connecteurs ou des types standards.

Les principes généraux d'ajout de constructions logiques sont les suivants :

– chaque **construction** se caractérise par un ensemble de constructeurs de valeurs et un ensemble de constructeurs de contextes d'évaluation linéaires ;
– les constructeurs de valeurs interagissent avec les constructeurs de contextes linéaires pour se décomposer en interactions plus élémentaires ;
– des types spécifiques peuvent être associés aux constructeurs pour garantir que les interactions se passent entre constructeurs associées à la même construction ;
– des règles d'η-conversion seront associées à chaque construction, tant du côté termes que du côté contexte d'évaluation : leur rôle est de révéler l'aspect calculatoire implicite des variables relativement à la construction considérée ; autrement dit, elles caractérisent le comportement observationnel des variables dans des interactions propres à la construction considérée en affirmant l'unicité de la manière de construire un objet de cette construction ;
– des restrictions de syntaxe correspondantes, conséquences de l'η-conversion et spécifiques aux restrictions appel par nom et appel par valeur.

Danos, Joinet et Schellinx [DJS97] font une étude approfondie des plongements de LK et de certaines restrictions dans la logique linéaire [Gir87]. De manière générale, l'ajout, à la restriction appel par nom (resp. appel par valeur) du système $\mu\tilde{\mu}$, de la forme la plus générale de chacune des constructions logiques correspond à la version unicolore t (resp. q) de LK^{tq} (c'est-à-dire LK) chez Danos, Joinet et Schellinx. En revanche, les ajouts des formes restreintes, spécifiques à chaque sémantique d'évaluation, semblent correspondre aux (versions unicolores des) restrictions LK^{η} que Danos, Joinet et Schellinx montrent précisément être conséquences de l'η-expansion.

Il semble que certaines de nos règles η soient appelées θ chez Danos, Joinet et Schellinx. C'est une terminologie à étudier.

3.1.1 Généralités sur l'η-conversion

C'est avant tout appliquées à des variables que les règles d'η-conversion sont pertinentes. Puisque les variables sont instanciables par n'importe quelle catégorie de terme (resp. contexte d'évaluation), linéaire ou pas, en appel par nom (resp. en appel par valeur), les règles d'η-conversion pourront avoir pour conséquence de transformer des expressions non linéaires en expressions linéaires : tout terme devient assimilable à une valeur en CBN et tout contexte d'évaluation devient assimilable à un contexte d'évaluation linéaire en CBV. C'est cette identification, très forte, qui induira des restrictions de syntaxe spécifiques à l'appel par nom et à l'appel par valeur.

Pour indiquer qu'une règle d'η-conversion s'applique aux variables de terme ou aux variables de contexte d'évaluation, à défaut d'une meilleure notation, on utilise respectivement les exposants R

(comme « Right ») et L (comme « Left »), par référence au côté du séquents où les termes et contextes d'évaluation sont typés, et ceci bien que l'η-conversion fasse du sens même dans le calcul non typé).

Soit $x = \phi(x)$ une règle d'η-conversion de termes ($\phi(x)$ est une expression close dépendant de x). En appel par nom, on a

$$\begin{array}{lll} v & =_{\eta_\mu} & \mu\alpha.\langle v \| \alpha\rangle \\ & =_{\tilde{\mu}} & \mu\alpha.\langle v \| \tilde{\mu}x.\overline{\langle x \| \alpha\rangle}\rangle \\ & = & \mu\alpha.\langle v \| \tilde{\mu}x.\overline{\langle \phi(x) \| \alpha\rangle}\rangle \\ & =_{\tilde{\mu}} & \mu\alpha.\langle \phi(v) \| \alpha\rangle \\ & =_{\eta_\mu} & \phi(v) \end{array}$$

si bien que la règle devient applicable à tout terme. En appel par valeur, le même raisonnement avec ($\tilde{\mu}_v$) au lieu de ($\tilde{\mu}$) permet de dériver $V = \phi(V)$.

Parallèlement, si $\alpha = \phi(\alpha)$ est une règle d'η-conversion de contextes d'évaluation, la règle $E = \phi(E)$ est dérivable en appel par nom, et la règle $e = \phi(e)$ est dérivable en appel par valeur.

On utilise alors la lettre n en indice d'un nom de règle d'η-conversion pour désigner la généralisation de cette règle au cas d'un terme arbitraire si la règle s'applique aux variables de terme, ce qui est dérivable en appel par nom, et au cas d'un contexte d'évaluation linéaire arbitraire si la règle s'applique aux variables de contexte d'évaluation.

Parallèlement, on utilise la lettre v pour désigner la généralisation d'une règle d'η-conversion qui est valide en appel par valeur.

En présence de plusieurs constructions, chacune équipée avec sa ou ses règles d'η-conversion, le calcul devient dégénéré s'il n'est pas en plus équipé d'un système de typage. En effet, en l'absence de contraintes de type, tout constructeur d'une construction devient égal à tout autre constructeur d'une autre construction. Dans un cadre non typé, il convient donc d'orienter les règles η dans le sens de la simplification de l'expression.

En présence d'un seul type de construction (par exemple dans le calcul non typé avec constructeurs de l'implication), on peut toutefois imaginer aller dans le sens de l'expansion, mais c'est une option qui moralement conduit à des « formes normales » expansées à l'infini (typiquement des arbres de Nakajima).

Si l'on oriente les règles dans le sens de la simplification des expressions, on se retrouve face à une perte de confluence spécifique au extensions du système $\mu\tilde{\mu}$. Pour chaque cas de constructeur, les η-réductions associées nécessiteront donc de compléter le système de réduction afin de restaurer la confluence.

3.1.2 Classification des calculs étendus avec des constructions logiques

On dénotera chacune des constructions par le nom du connecteur logique qui lui est spontanément associé. Comme connecteurs logiques on étudiera entre autres

- l'implication, connecteur binaire dénoté par $\to$,
- la conjonction, connecteur binaire dénoté par $\wedge$,
- la disjonction, connecteur binaire dénoté par $\vee$,
- la soustraction, connecteur binaire dénoté par $-$,
- la négation, connecteur unaire dénoté par $\neg$,
- le vrai, connecteur constant dénoté par $\top$,
- le faux, connecteur constant dénoté par $\bot$,

À chacun des connecteurs binaires, on associera une construction composée d'une part d'un constructeur binaire (avec deux arguments) et d'autre part, au choix,

- d'un constructeur unique composé d'un lieur binaire, et appelé constructeur multiplicatif ;
- d'un constructeur unique composé d'un lieur unaire, et appelé constructeur multiplicatif asymétrique ;
- ou de deux constructions distinctes non liantes, appelées constructeurs additifs.

À chacun de ces constructeurs, on associera une ou deux règles de réduction caractérisant l'interaction avec le constructeur binaire :

- au constructeur multiplicatif, on associe une ou deux règles asymétriques notées avec l'indice $_m$ (si le connecteur est symétrique) et les exposants ^{1} ou ^{2} ;
- aux constructeurs multiplicatifs asymétriques, on associe une règle écrite sans annotation ou avec l'exposant $^\perp$ pour les connecteurs binaires asymétriques ($\to$ et $-$), et écrite avec les indices $_{m_1}$ ou $_{m_2}$ pour les connecteurs binaires symétriques ($\wedge$ et $\vee$) ;

– aux constructeurs additifs, on associe une règle notée avec l'indice $_a$ et les exposants ^{1} ou ^{2}.

On notera les constructions sur la base de leurs règles de réduction associées (sauf pour les constructeurs constants qui n'ont pas de règles de réduction associée et que l'on notera par le nom du connecteur associé). Si la construction associée à un certain connecteur est basée sur des constructeurs additifs, on la représentera par le nom du connecteur associé indexé par la lettre a. Si la construction est basée sur un constructeur multiplicatif, on la représentera par le nom du connecteur indexé par la lettre m, et, si une seule des deux règles possibles est présente, par un exposant ^{1} ou ^{2}. Finalement, si la construction est basée sur un constructeur multiplicatif asymétrique, on la représentera par le nom de l'unique règle associée.

Si une construction est restreinte, comme conséquence d'une propriété d'η-conversion, on ajoutera la lettre η en exposant du nom de la construction.

Pour toute extension du sous-système $\mu\tilde{\mu}$ (ou l'une de ses restrictions CBN ou CBV) avec un ensemble de constructions, on notera le calcul résultat $\mu\tilde{\mu}$ en ajoutant en exposant la liste des constructions considérées. Pour un connecteur donné, la forme additive et l'un ou l'autre des formes multiplicatives de la construction peuvent (virtuellement) coexister auquel cas le nom du connecteur apparaît plusieurs fois, chacune avec son annotation spécifique. Par exemple, le calcul

$$\mu\tilde{\mu}_n^{\rightarrow^\eta \rightarrow_a^\eta \wedge_m \top}$$

désigne la restriction appel par nom du calcul $\mu\tilde{\mu}$ étendu avec les constructions $\rightarrow$, $\rightarrow_a$, $\wedge_m$ (avec ses deux règles de réduction) et $\top$, sachant en particulier, que les constructions $\rightarrow$, $\rightarrow_a$ partagent le même constructeur de contexte d'évaluation mais présentent deux styles distincts de constructeurs de termes. De plus, le constructeur que $\rightarrow$ et $\rightarrow_a$ partagent est restreint par les conséquences de l'η-conversion en appel par nom.

3.2 Implication (calcul $\overline{\lambda}\mu\tilde{\mu}$)

3.2.1 Abstraction et contexte applicatif

Dans cette section, nous considérons l'extension du sous-système $\mu\tilde{\mu}$ avec la construction **implication**. Celle-ci se décompose en un constructeur de valeur, l'**abstraction**, et en un constructeur de contextes d'évaluation linéaires, qui remplace la notion d'application du λ-calcul usuel, et que l'on appelle constructeur de **contextes (d'évaluation) applicatifs**.

Nous obtenons ainsi le $\mu\tilde{\mu}^{\rightarrow}$ (appelé calcul $\overline{\lambda}\mu\tilde{\mu}$ dans [CH00]) dont on caractérisera la version non-déterministe et les restrictions à l'appel par nom et à l'appel par valeur, ainsi que la restriction intuitionniste.

Nous montrerons qu'à certaines conditions les versions CBN des calculs $\overline{\lambda}\mu\tilde{\mu}$ et $\lambda\mu$ sont isomorphes. Le cas de l'appel par valeur est plus complexe (plus complexe à étudier dans le calcul $\lambda\mu$ s'entend) et nous ne le traiterons pas. De même pour les variantes non-déterministes de $\lambda\mu$.

L'ajout de l'abstraction et de l'application au sous-système $\mu\tilde{\mu}$ avec implication se caractérise par l'extension de grammaire suivante :

$$\begin{array}{lcl} V_c & ::= & \ldots \mid \lambda x.v \\ E_c & ::= & \ldots \mid v \cdot e \end{array}$$

Le constructeur de terme $\lambda x.v$ est standard. Quant à la construction $v \cdot e$, elle forme le contexte d'évaluation qui consiste à appliquer v puis à placer le résultat obtenu dans le contexte e. C'est la construction équivalente au contexte d'évaluation $e[\Box\, v]$ pour le λ-calcul traditionnel, c'est-à-dire à l'instanciation du trou du contexte e par le contexte $\Box\, v$.

La règle de réduction associée[1] est

$$(\rightarrow) \quad \langle \lambda x.v \| v' \cdot e\rangle \quad \rightarrow \quad \langle v' \| \tilde{\mu}x.\langle \overline{v \| e}\rangle\rangle$$

Les connexions entre le calcul $\overline{\lambda}\mu\tilde{\mu}$ et le calcul $\lambda\mu$ sont étudiées en section 3.13.

[1] Cette règle est notée $(\rightarrow')$ dans [CH00] tandis que le nom $(\rightarrow)$ est utilisé pour la règle qui sera notée $(\rightarrow^\beta)$ dans la section 3.2.5. Cette interversion de nom est justifiée par le fait que nous considérons en fait la nouvelle règle $(\rightarrow)$ comme plus primitive

3.2.2 η-conversion

Le rôle de l'η-conversion pour les connecteurs est d'identifier (si possible) une expression atomique (une variable) avec une expression construite de même comportement observationnel.

La conversion entre une variable de terme et une abstraction correspond à la règle standard d'η-conversion du λ-calcul, modulo une μ-expansion rendue nécessaire par la syntaxe du $\overline{\lambda}\mu\tilde{\mu}$. Par uniformité de notations, nous nommons cette conversion $\eta^R_{\rightarrow}$.

$$(\eta^R_{\rightarrow}) \qquad y \quad = \quad \lambda x.\mu\alpha.\langle y \| x \cdot \alpha\rangle \quad x \text{ et } y \text{ distincts}$$

Dans le cas de la restriction CBN, par conjugaison avec la conversion η_μ et la règle $(\tilde{\mu})$, on obtient la généralisation

$$(\eta^R_{\rightarrow n}) \qquad v \quad = \quad \lambda x.\mu\alpha.\langle v \| x \cdot \alpha\rangle \quad x \text{ et } \alpha \text{ non libres dans } v$$

Notons au passage que le seul cas intéressant est celui d'un $\mu\alpha.c$ car si v est une abstraction, la conversion est équivalente à une réduction ($\rightarrow$). On a retrouvé l'équivalent de l'η-conversion standard du λ-calcul usuel.

Dans le cas de la restriction CBV, la généralisation n'est dérivable et valide que pour les valeurs (car sinon le non-déterminisme réapparaît). Par composition avec $(\tilde{\mu}_v)$, on obtient la généralisation suivante, équivalente à l'η-conversion standard du λ-calcul par valeur :

$$(\eta^R_{\rightarrow v}) \qquad V \quad = \quad \lambda x.\mu\alpha.\langle V \| x \cdot \alpha\rangle \quad x \text{ et } \alpha \text{ non libres dans } V$$

À la différence du λ-calcul, le calcul $\mu\tilde{\mu}^{\rightarrow}$ accepte aussi une η-conversion entre variables de contexte d'évaluation et contexte applicatif. La règle est la suivante :

$$(\eta^L_{\rightarrow}) \qquad \gamma \quad = \quad \mu\beta.\langle \lambda x.\mu\alpha.\overline{\langle x \| \beta\rangle} \| \gamma\rangle \cdot \tilde{\mu}y.\langle \lambda x.y \| \gamma\rangle \quad \beta \text{ et } \gamma \text{ distincts}$$

Symétriquement au cas des variables de terme, on a dans le cas CBN la généralisation suivante :

$$(\eta^L_{\rightarrow n}) \qquad E \quad = \quad \mu\beta.\langle \lambda x.\mu\alpha.\overline{\langle x \| \beta\rangle} \| E\rangle \cdot \tilde{\mu}y.\langle \lambda x.y \| E\rangle \quad \beta \text{ et } y \text{ non libres dans } E$$

Quant au cas CBV, on a la généralisation :

$$(\eta^L_{\rightarrow n}) \qquad e \quad = \quad \mu\beta.\langle \lambda x.\mu\alpha.\overline{\langle x \| \beta\rangle} \| e\rangle \cdot \tilde{\mu}y.\langle \lambda x.y \| e\rangle \quad \beta \text{ et } y \text{ non libres dans } e$$

Notons aussi que cette règle, contrairement à $(\eta^R_{\rightarrow})$ n'est pas compatible avec la restriction intuitionniste car la variable α n'y est pas liée linéairement.

Proposition 18

- *Les règles $(\eta^L_{\rightarrow n})$ et $(\eta^R_{\rightarrow})$ font partie de la clôture observationnelle de $=_n$.*
- *Les règles $(\eta^L_{\rightarrow})$ et $(\eta^R_{\rightarrow v})$ font partie de la clôture observationnelle de $=_v$.*

Nous étudions dans la section suivante les conséquences de ces η-conversions sur la syntaxe des restrictions CBN et CBV, mais nous donnons d'abord quelques propriétés de l'η-conversion en appel par nom, en particulier que $(\eta^R_{\rightarrow n})$ est dérivable de $(\eta^L_{\rightarrow n})$. Une analyse plus approfondie, d'une part de la force de $(\eta^L_{\rightarrow})$, et d'autre part des propriétés de l'η-réduction en appel par valeur reste à faire.

3.2.3 Conséquences de l'η-conversion pour les termes en appel par nom

On a les propriétés suivantes.

Proposition 19 (Formes généralisées de l'η-conversion)

$$\begin{array}{lll} \lambda x.\mu\alpha.c[\alpha \leftarrow x \cdot \alpha] & =_{\eta^R_{\rightarrow n}\mu_n} \mu\alpha.c & x \textit{ non libre dans } c \\ \lambda x.\mu\_.c & =_{\eta^R_{\rightarrow n}\mu_n} \mu\_.c & x \textit{ non libre dans } c \\ \lambda x_1.\ldots.\lambda x_n.\mu\alpha.c[\alpha \leftarrow x_1 \cdot \ldots \cdot x \cdot \alpha] & =_{\eta^R_{\rightarrow n}\mu_n} \mu\alpha.c & x_1, \ldots, x_n \textit{ non libres dans } c \end{array}$$

En particulier, la première équation est l'équivalent pour le $\overline{\lambda}\mu\tilde{\mu}$ de la règle (ν) du calcul $\lambda\mu$ de David et Py [DP01].

Preuve: La première propriété est directe (cf David et Py [DP01]) :

$$\begin{array}{lll} \lambda x.\mu\alpha.c[\alpha \leftarrow x \cdot \alpha] & =_{\mu_n} & \lambda x.\mu\alpha.\langle \mu\alpha.c \| x \cdot \alpha \rangle \\ & =_{\eta^R_{\to n}} & \mu\alpha.c \end{array}$$

Quant à la seconde, c'est un cas particulier de la première, et la troisième, une généralisation. ∎

Proposition 20 (Décomposition additive de l'abstraction)

$$\begin{array}{lll} \mu\beta.\langle \lambda x.\mu\_.\overline{\langle \lambda\_.\mu\alpha.c \| \beta \rangle} \| \beta \rangle & =_{\eta^R_{\to n}\mu_n\tilde{\mu}\to} & \lambda x.\mu\alpha.c \\ \mu\beta.\langle \lambda\_.\mu\alpha.\overline{\langle \lambda x.\mu\_.c \| \beta \rangle} \| \beta \rangle & =_{\eta^R_{\to n}\mu_n\tilde{\mu}\to} & \lambda x.\mu\alpha.c \\ \langle \lambda x.\mu\_.\overline{\langle \lambda\_.\mu\alpha.c \| E \rangle} \| E \rangle & =_{\eta^R_{\to n}\mu_n\tilde{\mu}\to} & \langle \lambda x.\mu\alpha.c \| E \rangle \\ \langle \lambda\_.\mu\alpha.\overline{\langle \lambda x.\mu\_.c \| E \rangle} \| E \rangle & =_{\eta^R_{\to n}\mu_n\tilde{\mu}\to} & \langle \lambda x.\mu\alpha.c \| E \rangle \end{array}$$

Preuve: On a

$$\begin{array}{lll} \mu\beta.\langle \lambda x.\mu\_.\langle \overline{\lambda\_.\mu\alpha.c \| \beta} \rangle \| \beta \rangle & =_{\eta^R_{\to n}} & \lambda x.\mu\alpha.\langle \mu\beta.\langle \overline{\lambda x.\mu\_.\langle \overline{\lambda\_.\mu\alpha.c \| \beta} \rangle \| \beta} \rangle \| x \cdot \alpha \rangle \\ & =_{\mu_n} & \lambda x.\mu\alpha.\langle \lambda x.\mu\_.\langle \overline{\lambda\_.\mu\alpha.c \| x \cdot \alpha} \rangle \| x \cdot \alpha \rangle \\ & =_{\to\tilde{\mu}\mu_n} & \lambda x.\mu\alpha.c \end{array}$$

La deuxième équation se prouve de la même manière et les deux dernières équations s'obtiennent alors des premières par interaction avec E et réduction μ_n. ∎

Proposition 21 *En appel par nom, $\eta^R_{\to}$ implique $\eta^L_{\to}$*

Preuve: On a

$$\begin{array}{lll} \mu\beta.\langle \lambda x.\mu\_.\langle \overline{x \| \beta} \rangle \| \gamma \rangle \cdot \tilde{\mu}y.\langle \lambda\_.y \| \gamma \rangle & =_{\eta_{\tilde{\mu}}} & \tilde{\mu}z.\langle z \| \mu\beta.\underline{\langle \overline{\lambda x.\mu\_.\langle \overline{x \| \beta} \rangle \| \gamma} \rangle \cdot \tilde{\mu}y.\langle \overline{\lambda\_.y \| \gamma} \rangle \rangle} \\ & =_{\mu_n} & \tilde{\mu}z.\langle \mu\alpha.\langle z \| \mu\beta.\underline{\langle \overline{\lambda x.\mu\_.\langle \overline{x \| \beta} \rangle \| \alpha} \rangle \cdot \tilde{\mu}y.\langle \overline{\lambda\_.y \| \alpha} \rangle \rangle \rangle \| \gamma \rangle} \\ & =_{\eta^R_{\to}\mu_n} & \tilde{\mu}z.\langle \lambda x.\mu\alpha.\langle \underline{z \| \mu\beta.\langle \overline{\lambda x.\mu\_.\langle \overline{x \| \beta} \rangle \| x} \cdot \alpha \rangle} \cdot \tilde{\mu}y.\langle \overline{\lambda\_.y \| x \cdot \alpha} \rangle \rangle \rangle \| \gamma \rangle \\ & =_{\to\mu_n} & \tilde{\mu}z.\langle \lambda x.\mu\alpha.\langle \underline{z \| \mu\beta.}\langle \overline{x \| \beta} \rangle \cdot \tilde{\mu}y.\langle \overline{y \| \alpha} \rangle \rangle \rangle \| \gamma \rangle \\ & =_{\eta_\mu\eta_{\tilde{\mu}}} & \tilde{\mu}z.\langle \lambda x.\mu\alpha.\langle \overline{z} \| x \cdot \alpha \rangle \| \gamma \rangle \\ & =_{\eta^R_{\to}} & \tilde{\mu}z.\langle z \| \gamma \rangle \\ & =_{\eta_{\tilde{\mu}}} & \gamma \end{array}$$

∎

3.2.4 La restriction de syntaxe pour l'implication en appel par nom

En présence de $(\eta^R_{\to})$, on a

$$\begin{array}{lll} v \cdot \tilde{\mu}x.c & =_{\eta_{\tilde{\mu}}} & \tilde{\mu}y.\langle y \| v \cdot \tilde{\mu}x.c \rangle \\ & =_{\eta^R_{\to}} & \tilde{\mu}y.\langle \lambda z.\mu\alpha.\langle \overline{y \| z \cdot \alpha} \rangle \| v \cdot \tilde{\mu}x.c \rangle \\ & =_{\to} & \tilde{\mu}y.\langle v \| \tilde{\mu}z.\langle \overline{\mu\alpha.\langle \overline{y} \| z \cdot \alpha \rangle \| \tilde{\mu}x.c} \rangle \rangle \\ & =_{\tilde{\mu}} & \tilde{\mu}y.\langle v \| \tilde{\mu}z.c[x \leftarrow \mu\alpha.\langle \overline{y} \| z \cdot \alpha \rangle] \rangle \\ & =_{\tilde{\mu}} & \tilde{\mu}y.c[x \leftarrow \mu\alpha.\langle y \| v \cdot \alpha \rangle] \end{array}$$

Toute expression contenant des sous-contextes de la forme $v \cdot \tilde{\mu}x.c$ est donc équivalente à une expression qui ne contient que des sous-contextes d'évaluation de la forme $v \cdot E$. De plus ce genre de contexte est stable par substitution CBN. On peut donc restreindre le calcul en appel par nom à la syntaxe suivante :

Syntaxe restreinte de l'implication en appel par nom

$$\begin{array}{lll} V_c & ::= & \ldots \mid \lambda x.v \\ E_c & ::= & \ldots \mid v \cdot E \end{array}$$

On appelle ce calcul contraint le calcul $\mu_n\tilde{\mu}^{\to^\eta}$ où la restriction obtenue procède du même mécanisme que le passage de la version unicolore t de LK^{tq} à la version unicolore t de LK^η dans Danos, Joinet et Schellinx [DJS97] (normalisation d'une coupure obtenue par η-expansion d'une règle d'axiome). La section suivante analyse comment poursuivre la simplification sans perte d'expressivité.

Notons que le calcul $\mu_n\tilde{\mu}^{\to^\eta}$ est nommé calcul $\overline{\lambda}\mu\tilde{\mu}_T$ dans [CH00].

3.2.5 Le calcul en appel par nom restreint $\overline{\lambda}\mu_n$

L'opérateur $\tilde{\mu}$ n'apparaissant plus dans les commandes en forme normale du calcul $\mu_n\tilde{\mu}^{\to^\eta}$, on peut aller plus loin en supprimant $\tilde{\mu}$ de la syntaxe et en combinant $(\to)$ et $(\tilde{\mu})$ en une unique règle. Nous obtenons ce que nous appelons le calcul $\overline{\lambda}\mu_n$ (cf [CH00]) :

Calcul $\overline{\lambda}\mu_n$

$$\begin{array}{lcl} c & ::= & \langle v \| E \rangle \\ v & ::= & \mu\alpha.c \mid x \mid \lambda x.v \\ E & ::= & \alpha \mid v \cdot E \end{array}$$

Règles de réduction

$$\begin{array}{llcl} (\mu_n) & \langle \mu\alpha.c \| E \rangle & \xrightarrow{h} & c[\alpha \leftarrow E] \\ (\to^\beta) & \langle \lambda x.v \| v' \cdot E \rangle & \xrightarrow{h} & \langle v[x \leftarrow v'] \| E \rangle \end{array}$$

Dans cette restriction, la règle d'η-conversion $\eta_{\tilde{\mu}}$ disparaît et la règle d'η-conversion η_μ est orientable sans souci dans le sens de la décroissance de la taille de l'expression. En revanche, l'orientation de la règle d'η-conversion $(\eta^R_{\to n})$ pose problème. Si on l'oriente dans le sens de la décroissance de la taille de l'expression, on perd la confluence car $\lambda x.\mu\alpha.\langle \mu\beta.c \| x \cdot \alpha \rangle$ se réduit par $(\eta^R_{\to n})$ en $\mu\beta.c$ et se réduit par (μ_n) en $\lambda x.\mu\alpha.c[\beta \leftarrow x \cdot \alpha]$ sans possibilité de convergence (cf David et Py [DP01] pour une analyse du problème). En revanche, si l'on oriente $(\eta^R_{\to n})$ dans le sens de l'expansion, ce que nous faisons ci-dessous, la confluence est raisonnablement conjecturable mais on perd la notion de forme normale puisque tout terme est expansable.

Règles d'η-réduction

$$\begin{array}{llcll} (\eta_\mu) & \mu\alpha.\langle v \| \alpha \rangle & \xrightarrow{h} & v & \alpha \text{ non libre dans } v \\ (\eta^R_{\to n}) & v & \xrightarrow{h} & \lambda x.\mu\alpha.\langle v \| x \cdot \alpha \rangle & x \text{ et } \alpha \text{ non libres dans } v \end{array}$$

Notons aussi que cette restriction entraîne une légère perte d'expressivité au niveau des contextes d'évaluation puisque les expressions normales de la forme $\tilde{\mu}x.c$ (à l'exception de celle de la forme $\tilde{\mu}x.\langle x \| v \rangle$ avec x non libre dans c) n'ont plus d'équivalent.

Relecture du calcul $\overline{\lambda}\mu_n$ avec les notations du λ-calcul

Il peut être instructif de relire le calcul $\overline{\lambda}\mu_n$ avec une syntaxe proche de celle du λ-calcul en intégrant la catégorie syntaxique des contextes d'évaluation à celle des commandes.

Calcul $\overline{\lambda}\mu_n$ en notation « λ-calcul »

$$\begin{array}{lcll} c & ::= & [\beta]v\, v_1 \ldots v_n & \quad (\text{pour } n \geq 0) \\ v & ::= & \mu\alpha.c \mid x \mid \lambda x.v & \end{array}$$

Règles de réduction

$$\begin{array}{llcl} (\mu_n) & [\beta]((\mu\alpha.c)\, v_1 \ldots v_n) & \xrightarrow{h} & c[\alpha \leftarrow [\beta](\Box\, v_1 \ldots v_n)] \\ (\to^\beta) & [\beta]((\lambda x.v)\, v'\, v_1 \ldots v_n) & \xrightarrow{h} & [\beta](v[x \leftarrow v']\, v_1 \ldots v_n) \end{array}$$

où $c[\alpha \leftarrow [\beta]\Box\, v_1 \ldots v_n]$ désigne la substitution sans capture des sous-termes de la forme $[\alpha](v\, v'_1 \ldots v'_{n'})$ dans c par $[\beta](v\, v'_1 \ldots v'_{n'}\, v_1 \ldots v_n)$.

Règles d'η-réduction

$$\begin{array}{llcll} (\eta_\mu) & \mu\alpha.[\alpha]v & \xrightarrow{h} & v & \alpha \text{ non libre dans } v \\ (\eta^R_{\to n}) & \lambda x.\mu\alpha.[\alpha](v\, x) & \xrightarrow{h} & v & x \text{ et } \alpha \text{ non libres dans } v \end{array}$$

On s'aperçoit que les règles β et η du λ-calcul usuel sont tout près. Si on change le point de vue de telle sorte que $(\lambda x.v)\, v'$ soit considéré comme sous-terme de $[\beta]((\lambda x.v)\, v'\, v_1 \ldots v_n)$ (ce qu'il n'est pas dans le calcul $\overline{\lambda}\mu_n$), on peut simplifier la règle $(\to^\beta)$ en la règle standard

$$(\lambda x.v)\, v' \xrightarrow{h} v[x \leftarrow v']\,.$$

De même, à partir du moment où $v\,x$ est considéré comme un sous-terme de $[\alpha](v\,x)$, on peut, grâce à (η_μ), simplifier $(\eta^R_{\rightarrow n})$ en la règle standard

$$\lambda x.(v\,x) \xrightarrow{h} v \qquad \text{pour } x \text{ non libre dans } v.$$

3.2.6 La restriction de syntaxe pour l'implication en appel par valeur

En présence de $(\eta^R_{\rightarrow v})$, on a pour l'appel par valeur,

$$\begin{array}{lll}
\mu\beta.c\cdot e & =_{\eta_{\tilde\mu}} & \tilde\mu y.\langle y\|\mu\beta.c\cdot e\rangle \\
 & =_{\eta^R_{\rightarrow v}} & \tilde\mu y.\langle \lambda z.\mu\alpha.\overline{\langle y\|z\cdot\alpha\rangle}\|\mu\beta.c\cdot e\rangle \\
 & =_{\rightarrow} & \tilde\mu y.\langle \mu\beta.c\|\tilde\mu z.\langle\overline{\mu\alpha.\overline{\langle y\|z\cdot\alpha\rangle}}\|e\rangle\rangle \\
 & =_{\mu} & \tilde\mu y.\langle \mu\beta.c\|\tilde\mu z.\overline{\langle y\|z\cdot e\rangle}\rangle \\
 & =_{\mu} & \tilde\mu y.c[\beta\leftarrow \tilde\mu z.\langle y\|z\cdot e\rangle]
\end{array}$$

On en déduit que toute expression contenant des sous-contextes de la forme $\mu\alpha.c\cdot e$ est équivalente à une expression qui ne contient que des sous-contextes d'évaluation de la forme $V\cdot e$. De plus ce genre de contexte est stable par substitution CBV. On peut donc restreindre le calcul en appel par valeur à la syntaxe suivante que l'on appelle au choix $\mu\tilde\mu_v^{\rightarrow^\eta}$ (si l'on suit le schéma canonique de nommage) ou $\overline{\lambda}\mu\tilde\mu_Q$ (si l'on suit la terminologie de [CH00]).

Syntaxe restreinte de l'implication en appel par valeur

$$\begin{array}{lll}
V_c & ::= & \ldots \mid \lambda x.v \\
E_c & ::= & \ldots \mid V\cdot e
\end{array}$$

Comme pour le calcul $\mu_n\tilde\mu^{\rightarrow^\eta}$, la restriction obtenue procède du même mécanisme que le passage de la version unicolore q de LK^{tq} à la version unicolore q de LK^η dans Danos, Joinet et Schellinx [DJS97].

On trouvera par ailleurs des preuves de confluence (par la méthode des réductions parallèles) de la restriction appel par nom équipée de la règle $(\rightarrow^\beta)$ et de la restriction appel par valeur équipée de la règle $(\rightarrow)$ dans la thèse de Livakec [Lik05].

Résumé de la description du calcul $\mu\tilde\mu_v^{\rightarrow^\eta}$

Calcul $\mu\tilde\mu_v^{\rightarrow^\eta}$

$$\begin{array}{lll}
c & ::= & \langle v\|e\rangle \\
V & ::= & x \mid \lambda x.v \\
v & ::= & V \mid \mu\alpha.c \\
e & ::= & \alpha \mid V\cdot e \mid \tilde\mu x.c
\end{array}$$

Règles de réduction

$$\begin{array}{llcl}
(\rightarrow) & \langle \lambda x.v\|V\cdot e\rangle & \xrightarrow{h} & \langle V\|\tilde\mu x.\overline{\langle v\|e\rangle}\rangle \\
(\mu) & \langle \mu\alpha.c\|e\rangle & \xrightarrow{h}_v & c[\alpha\leftarrow e] \\
(\tilde\mu_v) & \langle V\|\tilde\mu x.c\rangle & \xrightarrow{h} & c[x\leftarrow V]
\end{array}$$

De plus, les règles d'η-conversion sont orientables en un système que l'on peut raisonnablement conjecturer confluent (notons que (η_μ) ne devient pertinent qu'en tête).

Règles d'η-réduction

$$\begin{array}{llcll}
(\eta_\mu) & \mu\alpha.\langle v\|\alpha\rangle & \xrightarrow{h} & v & \alpha \text{ non libre dans } v \\
(\eta_{\tilde\mu}) & \tilde\mu x.\langle x\|e\rangle & \xrightarrow{h} & e & x \text{ non libre dans } e \\
(\eta^R_{\rightarrow v}) & \lambda(x,\alpha).\langle V\|x\cdot\alpha\rangle & \xrightarrow{h} & V & x \text{ et } \alpha \text{ non libres dans } V
\end{array}$$

3.2.7 Le calcul en appel par valeur contraint $\overline{\lambda}\tilde{\mu}_v$

Par analogie avec le calcul $\overline{\lambda}\mu_n$ dans lequel $\tilde{\mu}$ a disparu, on peut définir un calcul $\overline{\lambda}\tilde{\mu}_v$ pour l'appel par valeur dans lequel l'opérateur μ ne figure plus explicitement. Ce calcul vient de [CH00].

Par application de (η_μ), on peut identifier tout terme de la forme $\lambda x.v$ avec un terme de la forme $\lambda x.\mu\alpha.c$, quitte à expanser v en $\mu\alpha.\langle v\|\alpha\rangle$ s'il n'est pas déjà sous la forme $\mu\alpha.c$. Si l'on accepte une légère perte d'expressivité pour les termes, du même ordre que la perte en expressivité pour les contextes d'évaluation dans le calcul $\overline{\lambda}\mu_n$, on peut utiliser l'abstraction double de la section 3.4 pour représenter tout sous-terme de la forme $\lambda x.\mu\alpha.c$ par $\lambda(x,\alpha).c$ et ainsi renoncer complètement à l'opérateur μ. La perte d'expressivité, c'est celle des expressions de la forme $\mu\alpha.c$ (type d'expression qui ne peut d'ailleurs exister qu'en tête dans les formes normales du calcul $\mu\tilde{\mu}_v^{\to^\eta}$).

Par ailleurs, l'argument d'une application étant nécessairement une valeur, on peut, comme dans le calcul $\overline{\lambda}\mu_n$, contracter les règles $(\to)$ et $(\tilde{\mu})$ en une unique règle $(\to_v^\beta)$ comparable à la β_v-réduction du λ-calcul usuel par valeur. Le calcul contraint est stable par réduction, nous l'appelons calcul $\overline{\lambda}\tilde{\mu}_v$:

Calcul $\overline{\lambda}\tilde{\mu}_v$

$$\begin{array}{lcl} c & ::= & \langle V\|e\rangle \\ V & ::= & x \mid \lambda(x,\alpha).c \\ e & ::= & \alpha \mid V\cdot e \mid \tilde{\mu}x.c \end{array}$$

Règles de réduction

$$\begin{array}{lllll} (\tilde{\mu}_v) & \langle V\|\tilde{\mu}x.c\rangle & \xrightarrow{h} & c[x\leftarrow V] \\ (\to_v^\beta) & \langle\lambda(x,\alpha).c\|V\cdot e\rangle & \xrightarrow{h} & c[x\leftarrow V][\alpha\leftarrow e] \end{array}$$

De plus, les règles d'η-conversion (hors η_μ qui a disparu) sont orientables en un système que l'on peut raisonnablement conjecturer confluent.

Règles d'η-réduction

$$\begin{array}{lllll} (\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x\|e\rangle & \xrightarrow{h} & e & x \text{ non libre dans } e \\ (\eta_{\to v}^R) & \lambda(x,\alpha).\langle V\|x\cdot\alpha\rangle & \xrightarrow{h} & V & x \text{ et } \alpha \text{ non libres dans } V \end{array}$$

Relecture du calcul $\overline{\lambda}\tilde{\mu}_v$ avec les notations du λ-calcul

À titre indicatif, on peut relire le calcul $\overline{\lambda}\tilde{\mu}_v$ dans une syntaxe proche de celle du λ-calcul (on s'inspire ici de la section 6 de [CH00]) en intégrant la catégorie syntaxique des contextes d'évaluation à l'intérieur de celle des commandes.

Calcul $\overline{\lambda}\tilde{\mu}_v$ en notation « λ-calcul »

$$\begin{array}{lcll} c & ::= & [\alpha](V\,V_1\ldots V_n) \mid \mathbf{let}\ x = V\,V_1\ldots V_n\ \mathbf{in}\ c & (\text{pour } n\geq 0) \\ V & ::= & x \mid \lambda(x,\alpha).c \end{array}$$

Règles de réduction

$$\begin{array}{llll} (\tilde{\mu}_v) & \mathbf{let}\ x = V\ \mathbf{in}\ c & \xrightarrow{h} & c[x\leftarrow V] \\ (\to_v^{\beta_1}) & [\beta]((\lambda(x,\alpha).c)\,V\,V_1\ldots V_n) & \xrightarrow{h} & c[x\leftarrow V][\alpha\leftarrow[\beta]\Box\,V_1\ldots V_n] \\ (\to_v^{\beta_2}) & \mathbf{let}\ x = (\lambda(x,\alpha).c)\,V\,V_1\ldots V_n\ \mathbf{in}\ c' & \xrightarrow{h} & c[x\leftarrow V][\alpha\leftarrow \mathbf{let}\ x = \Box\,V_1\ldots V_n\ \mathbf{in}\ c'] \end{array}$$

où α est choisi frais pour éviter une collision avec les variables de V_1 et la notation $c[\alpha\leftarrow[\beta]\Box\,V_1\ldots V_n]$ (resp. $c[\alpha\leftarrow \mathbf{let}\ x=\Box\,V_1\ldots V_n\ \mathbf{in}\ c']$) désigne la substitution sans capture des sous-termes de la forme $[\alpha](V\,V'_1\ldots V'_{n'})$ dans c par $[\beta](V\,V'_1\ldots V'_{n'}\,V_1\ldots V_n)$ (resp. $\mathbf{let}\ x = V\,V'_1\ldots V'_{n'}\,V_1\ldots V_n\ \mathbf{in}\ c'$).

Règles d'η-réduction

$$\begin{array}{lllll} (\eta_{\to v}^R) & \lambda(x,\alpha).[\alpha](V\,x) & \xrightarrow{h} & V & x \text{ et } \alpha \text{ non libres dans } V \end{array}$$

où la règle $(\eta_{\tilde{\mu}})$ ne fait plus de sens puisque la catégorie syntaxique des contextes d'évaluation a disparu.

Polarisation du calcul $\overline{\lambda}\tilde{\mu}_v$ et interprétation en termes de jeux

Par application de $(\eta_{\tilde{\mu}})$, on peut contraindre les contextes applicatifs à être de la forme $V \cdot \tilde{\mu}x.c$. On obtient un calcul dont les formes normales sont décrites par la syntaxe suivante :

Commandes normales du calcul $\overline{\lambda}\tilde{\mu}_v$ polarisé

$$c \quad ::= \quad \langle x \| \lambda(y,\alpha).c \cdot \tilde{\mu}z.c\rangle \mid \langle \lambda(x,\alpha).c \| \beta\rangle$$

Cette syntaxe de formes normales est à la base de l'interprétation de l'appel par valeur en termes de jeux de Honda et Yoshida [HY97].

Dans sa version intuitionniste, il n'y a qu'une unique variable de contexte d'évaluation que nous noterons $\star$. L'expression $\langle x \| \lambda(y,\star).c \cdot \tilde{\mu}z.c\rangle$ correspond à une attaque de la valeur liée à x par le joueur principal. Cette attaque est réalisée en annonçant détenir une valeur, en l'occurrence $\lambda(y,\star).c$, comme argument de x. Il y a deux réactions possibles de l'opposant à cette attaque : soit attaquer à son tour l'argument en associant une valeur à y, soit répondre par une valeur qui sera associée à la variable y.

L'expression $\langle \lambda(x,\star).c \| \star\rangle$ correspond à une réponse du joueur principal. Cette réponse annonce détenir une valeur, en l'occurrence $\lambda(y,\star).c$. L'unique réaction possible de l'opposant est alors d'attaquer en retour cette valeur en annonçant détenir une valeur comme instance de x.

On voit que la syntaxe fournit de manière directe une extension possible de cette notion de jeux au cas de l'appel par valeur classique. Il suffit de libéraliser la notion de réponse de telle sorte qu'une réponse ne réagit pas forcément à la dernière attaque mais à n'importe quelle attaque antérieure (à la manière de ce qui est fait pour l'appel par nom dans [Her97]).

3.2.8 Le calcul en appel par valeur $\overline{\lambda}_\eta\mu\tilde{\mu}_v$

À l'inverse, on peut ne pas restreindre la syntaxe du calcul $\mu\tilde{\mu}_v^{\rightarrow}$ mais prendre en compte l'η-conversion, orientée dans le sens de la simplification de l'expression. Le calcul obtenu n'est pas confluent puisque

$$\begin{array}{ccccc} & & \langle \lambda x.\mu\alpha.\overline{\langle y \| x \cdot \alpha\rangle} \| \mu\beta.c \cdot e\rangle & & \\ & \swarrow_{\eta_\rightarrow} & & \searrow_{\rightarrow\mu} & \\ \langle y \| v \cdot \tilde{\mu}z.c\rangle & & & & c[\beta \leftarrow \tilde{\mu}x.\langle y \| x \cdot e\rangle] \end{array}$$

avec le terme de gauche en forme normale de tête et celui de droite pas nécessairement réductible vers une commande de la forme $\langle y \| e\rangle$.

Il faut alors compléter le calcul avec une règle que nous nommons $(\rightarrow_\eta^\mu)$ et qui expanse les μ en partie gauche d'un contexte applicatif. On obtient alors le calcul suivant que nous conjecturons confluent.

Calcul $\overline{\lambda}_\eta\mu\tilde{\mu}_v$

$$\begin{array}{lcl} c & ::= & \langle v \| e\rangle \\ V & ::= & x \mid \lambda x.v \\ v & ::= & V \mid \mu\alpha.c \\ e & ::= & \alpha \mid v \cdot e \mid \tilde{\mu}x.c \end{array}$$

Règles de réduction

$$\begin{array}{llcll} (\mu) & \langle \mu\alpha.c \| e\rangle & \xrightarrow{h} & c[\alpha \leftarrow e] & \\ (\tilde{\mu}_v) & \langle V \| \tilde{\mu}y.c\rangle & \xrightarrow{h} & c[y \leftarrow V] & \\ (\rightarrow) & \langle \lambda x.v \| v' \cdot e\rangle & \xrightarrow{h} & \langle v' \| \tilde{\mu}x.\overline{\langle v \| e\rangle}\rangle & \\ (\rightarrow_\eta^\mu) & (\mu\alpha.c) \cdot e & \xrightarrow{h} & \tilde{\mu}y.c[\alpha \leftarrow \tilde{\mu}x.\langle y \| x \cdot e\rangle] & x \text{ et } y \text{ frais} \end{array}$$

Règles d'η-réduction

$$\begin{array}{llcll} (\eta_\mu) & \mu\alpha.\langle v \| \alpha\rangle & \xrightarrow{h} & v & \alpha \text{ non libre dans } v \\ (\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x \| e\rangle & \xrightarrow{h} & e & x \text{ non libre dans } e \\ (\eta^R_{\rightarrow v}) & \lambda x.\mu\alpha.\langle V \| x \cdot \alpha\rangle & \xrightarrow{h} & V & x \text{ et } \alpha \text{ non libres dans } v \end{array}$$

3.2.9 Typage de l'implication

Abstraction et constructeur de contexte applicatif se typent sans difficulté et produisent des expressions linéaires pour donner le système de types simples $LK_{\mu\tilde{\mu}}^{\rightarrow}$:

$$\frac{\Gamma, x : A \vdash v : B \,|\, \Delta)}{\Gamma \vdash \lambda x.v : A \to B \,;\, \Delta} \qquad \frac{\Gamma \vdash v : A \,|\, \Delta \qquad \Gamma \,|\, e : B \vdash \Delta}{\Gamma \,;\, v \cdot e : A \to B \vdash \Delta}$$

La nouvelle règle de réduction, ainsi que l'η-conversion, préservent le typage.

Proposition 22 *La règle de réduction ($\rightarrow$) préserve le typage, à la fois dans le sens réduction et dans le sens expansion.*

Les conversions ($\eta^R_\rightarrow$) et ($\eta^L_\rightarrow$) sont bien typées dans le sens perte d'information et ne sont bien typées dans le sens expansion que si l'expression d'origine a le type implication.

Notons que les restrictions de syntaxe liées aux restrictions CBN (resp. CBV) ont pour conséquence que la règle de construction de contextes applicatifs voit sa prémisse de droite (resp. de gauche) typée par un jugement d'expression linéaire. On obtient alors les systèmes $LK_{\mu_n\tilde{\mu}}^{\rightarrow}$ et $LK_{\mu\tilde{\mu}_v}^{\rightarrow}$.

Dans $LK_{\mu_n\tilde{\mu}}^{\rightarrow}$, toute forme normale est $\eta^R_\rightarrow$-convertible à une expression où chaque variable α est de type atomique. Ceci suggère d'orienter la règle $\eta^R_\rightarrow$ dans le sens d'une expansion de la taille du terme de telle sorte que toute commande normale est de l'une de ces deux formes

$$\begin{array}{l} \langle x \| \alpha \rangle \\ \langle x \| \lambda x_{11} \ldots x_{1n_1}.\mu\alpha_1.c_1 \cdot \ldots \cdot \lambda x_{m1} \ldots x_{mn_m}.\mu\alpha_m.c_m \cdot \alpha \rangle \end{array}$$

avec chaque α de type atomique[2].

3.2.10 Plongement dans le λ-calcul par passage de continuation

L'extension des traductions ^{n}, ^{nl}, ^{v} et ^{vl} de la section 2.10 au cas de l'implication nécessite (ou du moins est simplifié par) l'ajout d'un constructeur et d'un destructeur de paire au λ-calcul en appel par nom. L'extension, appliquée aux calculs restreints spécifiques de l'appel par nom et de l'appel par valeur, est obtenue comme suit :

$$\begin{array}{rclcrcl} (v \cdot E)^{nl} & \triangleq & (v^n, E^{nl}) & \qquad & (\lambda x.v)^n & \triangleq & \lambda(x,k).v^n\, k \\ (V \cdot e)^v & \triangleq & \lambda x.x\, V^{vl}\, e^v & & (\lambda x.v)^{vl} & \triangleq & \lambda x.v^v \end{array}$$

Au niveau du typage, on pose

$$\begin{array}{rcl} (A \to B)^n & \triangleq & \neg A^n \wedge B^n \\ (A \to B)^v & \triangleq & A^v \to \neg\neg B^v \end{array}$$

tout en gardant $A^n \triangleq A$ et $A^v \triangleq A$ sur les formules non construites. On vérifie la proposition

Proposition 23

On a, pour l'appel par nom
- $\Gamma \vdash v : A \,|\, \Delta$ *implique* $\neg\Gamma^n, \Delta^n \vdash v^v : \neg A^n$
- $\Gamma \,;\, E : A \vdash \Delta$ *implique* $\neg\Gamma^n, \Delta^n \vdash E^{ln} : A^n$
- $\Gamma \,|\, e : A \vdash \Delta$ *implique* $\neg\Gamma^n, \Delta^n \vdash e^n : \neg\neg A^n$
- $(c : \Gamma \vdash \Delta)$ *implique* $\neg\Gamma^n, \Delta^n \vdash c^n : \bot$

[2]Accessoirement, c'est la base pour une interprétation du calcul $\overline{\lambda}\mu\tilde{\mu}$ par nom en termes de jeux telle que décrite dans [Her97]. Une forme normale $\langle x \| \lambda x_{11} \ldots x_{1n_1}.\mu\alpha_1.c_1 \cdot \ldots \cdot \lambda x_{m1} \ldots x_{mn_m}.\mu\alpha_m.c_m \cdot \alpha \rangle$ (avec $m \geq 1$) correspond à une question du joueur sur la valeur ultime de l'application de x à ses arguments. Cette question ouvre la possibilité pour l'adversaire de questionner à son tour un des m arguments de x ou bien de répondre en donnant un résultat pour x. Une forme normale $\langle x \| \alpha \rangle$ correspond à une réponse par le joueur principal à une question de l'adversaire sur un argument dont la valeur était attendue sur le canal α (c'est-à-dire que cet argument se terminait par une expression de la forme $\mu\alpha.c$). Cette réponse n'ouvre en retour la possibilité d'aucun nouveau coup pour l'adversaire.

On a, pour l'appel par valeur
– $\Gamma \vdash V : A\,;\Delta$ *implique* $\Gamma^v, \neg\Delta^v \vdash V^{lv} : A^v$
– $\Gamma \vdash v : A \,|\, \Delta$ *implique* $\Gamma^v, \neg\Delta^v \vdash v^v : \neg\neg A^v$
– $\Gamma \,|\, e : A \vdash \Delta$ *implique* $\Gamma^v, \neg\Delta^v \vdash e^v : \neg A^v$
– $(c : \Gamma \vdash \Delta)$ *implique* $\Gamma^v, \neg\Delta^v \vdash c^v : \bot$

Notons que le typage de la traduction CPS CBN (qui vérifie A équivalent à $\neg A'^n$ où A' provient de A par négation des formules atomiques) correspond à la traduction de Streicher-Lafont [LRS93]. Notons aussi qu'on aurait pu tout aussi bien définir $(A \to B)^v$ comme $\neg(A^v \wedge \neg B^v)$.

3.3 De l'implication à la soustraction

3.3.1 Le calcul avec soustraction

L'implication est foncièrement asymétrique. Une étude de la dualité conduit donc naturellement à se poser la question du dual de l'implication, à savoir de la soustraction qui fut étudiée notamment par Rauszer [Rau74] et Crolard [Cro01]. On notera $A - B$ la soustraction des formules A et B.

Plus particulièrement, l'implication a une règle d'introduction binaire à deux prémisses et l'autre unaire à une prémisse. Les deux prémisses de la règle binaire sont duales puisque l'une concerne un terme tandis que l'autre concerne un contexte d'évaluation. En revanche la conclusion de la règle construit arbitrairement un terme alors qu'elle aurait pu construire, tout aussi arbitrairement, un contexte d'évaluation. C'est là qu'intervient la soustraction.

La soustraction aura une règle d'introduction binaire ayant les mêmes prémisses que la règle d'introduction binaire de l'implication (puisque ces prémisses sont invariantes par dualité) mais en introduction droite plutôt que gauche. On notera $v - e$ les objets ainsi construits. En ce qui concerne la règle d'introduction unaire, elle prendra, par dualité, un contexte d'évaluation e en prémisse et construira un autre contexte noté $\tilde{\lambda}\alpha.e$ par liaison d'une variable de contexte α. On abrégera $\tilde{\lambda}\alpha_1.\ldots\tilde{\lambda}\alpha_n.e$ en $\tilde{\lambda}\alpha_1\ldots\alpha_n.e$, voire en $\tilde{\lambda}\overrightarrow{\alpha}.e$. En résumé, on a :

$$\begin{array}{rcl} V_c & ::= & \ldots \,|\, v - e \\ E_c & ::= & \ldots \,|\, \tilde{\lambda}\alpha.e \end{array}$$

que l'on pourra typer avec les règles

$$\frac{\Gamma \,|\, e : A \vdash \alpha : B, \Delta}{\Gamma\,;\,\tilde{\lambda}\alpha.e : A - B \vdash \Delta} \qquad \frac{\Gamma \vdash v : A \,|\, \Delta \qquad \Gamma \,|\, e : B \vdash \Delta}{\Gamma \vdash v - e : A \to B\,;\,\Delta}$$

La règle de calcul associée est duale de celle de l'implication :

$$(-) \qquad \langle v - e' \| \tilde{\lambda}\alpha.e\rangle \;\;\to\;\; \langle \mu\alpha.\langle\overline{v\|e}\rangle \| e'\rangle$$

On note $\mu\tilde{\mu}^{-}$ le calcul $\mu\tilde{\mu}$ étendu avec la soustraction et $\mu\tilde{\mu}^{\to -}$ le système étendu avec la soustraction et l'implication. On note respectivement $LK^{-}_{\mu\tilde{\mu}}$ et $LK^{\to -}_{\mu\tilde{\mu}}$ les systèmes de types simples correspondants. On note $\mu_n\tilde{\mu}^{-}$, $\mu\tilde{\mu}^{-}_v$, $\mu_n\tilde{\mu}^{\to -}$ et $\mu\tilde{\mu}^{\to -}_v$ les restrictions respectives à l'appel par nom et à l'appel par valeur des calculs non typés.

3.3.2 Dualité dans le calcul $\mu\tilde{\mu}^{\to -}$

La soustraction permet d'étendre la dualité du sous-système $\mu\tilde{\mu}$ au cas de l'implication :

$$\begin{array}{rcl} (v \cdot e)^\circ & \triangleq & e^\circ - v^\circ \\ (\lambda x.v)^\circ & \triangleq & \tilde{\lambda}\alpha_x.v^\circ \\ \\ (v - e)^\circ & \triangleq & e^\circ \cdot v^\circ \\ (\tilde{\lambda}\alpha.e)^\circ & \triangleq & \lambda x_\alpha.e^\circ \end{array}$$

3.3.3 Règles d'η-conversion et restrictions induites

Sur le modèle de l'implication, et via le principe de dualité, on peut caractériser les règles d'η-conversion spécifiques de la soustraction.

$$\begin{array}{lcll}
(\eta^L_-) & e & = & \tilde{\lambda}\alpha.\tilde{\mu}x\langle x - \alpha \| e\rangle \\
(\eta^L_{-n}) & E & = & \tilde{\lambda}\alpha.\tilde{\mu}x\langle x - \alpha \| E\rangle \\
(\eta^R_-) & v & = & \mu\beta.\langle v \| \tilde{\lambda}\alpha.\beta\rangle - \tilde{\mu}y.\langle v \| \tilde{\lambda}\alpha.\tilde{\mu}x.\langle\overline{y \| \alpha}\rangle\rangle \\
(\eta^R_{-v}) & V & = & \mu\beta.\langle V \| \tilde{\lambda}\alpha.\beta\rangle - \tilde{\mu}y.\langle V \| \tilde{\lambda}\alpha.\tilde{\mu}x.\langle\overline{y \| \alpha}\rangle\rangle
\end{array}$$

Ces règles d'η-conversion induisent le même genre de restriction sur les fragments CBN et CBV.

Syntaxe restreinte de la soustraction en appel par nom ($\mu_n\tilde{\mu}^{-^\eta}$)

$$\begin{array}{lcl}
V_c & ::= & \ldots \mid v - E \\
E_c & ::= & \ldots \mid \tilde{\lambda}\alpha.e
\end{array}$$

Syntaxe restreinte de la soustraction en appel par valeur ($\mu\tilde{\mu}_v^{-^\eta}$)

$$\begin{array}{lcl}
V_c & ::= & \ldots \mid \tilde{\lambda}\alpha.v \\
E_c & ::= & \ldots \mid V - e
\end{array}$$

On vérifie en particulier que la dualité $\circ$ est une dualité entre $\mu_n\tilde{\mu}^{\to -}$ et $\mu\tilde{\mu}_v^{\to -}$, ainsi qu'entre $\mu_n\tilde{\mu}^{\to^\eta -^\eta}$ et $\mu\tilde{\mu}_v^{\to^\eta -^\eta}$.

3.3.4 Plongement dans le λ-calcul par passage de continuation

Le plongement des constructeurs restreints dans le λ-calcul par passage de continuation s'obtient par :

$$\begin{array}{lcl@{\qquad}lcl}
(V - e)^{vl} & \triangleq & (e^v, V^{vl}) & (\tilde{\lambda}\beta.e)^v & \triangleq & \lambda(x_\beta, y).e^v\, y \\[1em]
(v - E)^n & \triangleq & \lambda x.x\, E^{nl}\, v^n & (\tilde{\lambda}\alpha.e)^{nl} & \triangleq & \lambda x_\alpha.e^n
\end{array}$$

pour une traduction sur les types donnée par

$$\begin{array}{lcl}
(A - B)^n & \triangleq & B^n \to \neg\neg A^n \\
(A - B)^v & \triangleq & \neg B^v \wedge A^v
\end{array}$$

En particulier, en étendant la dualité aux types par

$$\begin{array}{lcl}
(A \to B)^\circ & \triangleq & B^\circ - A^\circ \\
(A - B)^\circ & \triangleq & B^\circ \to A^\circ\,,
\end{array}$$

on a les propriétés suivantes, sur les types, expressions et expressions linéaires :

$$\begin{array}{lcl}
{}^{vl} & = & {}^\circ \circ {}^{nl} \\
{}^{nl} & = & {}^\circ \circ {}^{vl} \\
{}^{v} & = & {}^\circ \circ {}^{n} \\
{}^{n} & = & {}^\circ \circ {}^{v}
\end{array}$$

3.3.5 Soustraction et restriction intuitionniste

En présence de la soustraction, l'invariant que les commandes et contextes d'évaluation avaient une et une seule variable libre de contexte et les termes aucune est remis en question.

La soustraction intuitionniste a été étudiée par Crolard [Cro01]. Notre définition du fragment intuitionniste (cf section 2.13) est-elle compatible avec l'analyse de Crolard ?

3.4 Implication et Modus Tollens

L'arbitraire des règles d'implication ne se situe pas que dans le côté d'introduction choisi (menant à la soustraction si on choisit l'autre côté), il se situe aussi dans la catégorie syntaxique de la prémisse de la règle unaire.

On peut par exemple s'intéresser à une abstraction qui lie aussi une variable de contexte, comme suit :

$$\begin{array}{lcl} V_c & ::= & \ldots \mid \lambda(x,\alpha).c \\ E_c & ::= & \ldots \mid v \cdot e \end{array}$$

Pour cette **abstraction double**, on a alors le choix entre les deux règles de réduction suivantes :

$$\begin{array}{llcl} (\to_1) & \langle \lambda(x,\alpha).c \| v \cdot e \rangle & \to & \langle v \| \tilde{\mu}x.\langle \overline{\mu\alpha.c \| e} \rangle \rangle \\ (\to_2) & \langle \lambda(x,\alpha).c \| v \cdot e \rangle & \to & \langle \mu\alpha.\langle \overline{v \| \tilde{\mu}x.c} \rangle \| e \rangle \end{array}$$

Malgré le côté asymétrique de ces règles, toutes deux se comportent de manière identique que ce soit en appel par nom ou en appel par valeur. De fait, comme conséquence de la proposition 4, on a :

Proposition 24 (Indifférence de l'ordre de séquentialisation de l'abstraction double)

– $c =_{\to_1 \mu_n \tilde{\mu}} c'$ *ssi* $c =_{\to_2 \mu_n \tilde{\mu}} c'$
– $c =_{\to_1 \mu \tilde{\mu}_v} c'$ *ssi* $c =_{\to_2 \mu \tilde{\mu}_v} c'$

Le typage correspondant est

$$\frac{c : (\Gamma, x : A \vdash \alpha : B, \Delta)}{\Gamma \vdash \lambda(x,\alpha).c : A \to B \,;\, \Delta} \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \,;\, v \cdot e : A \to B \vdash \Delta}$$

On peut aussi exiger que l'abstraction ne lie qu'une variable de contexte, et aucune variable de terme, comme suit :

$$\begin{array}{lcl} V_c & ::= & \ldots \mid \lambda\beta.e \\ E_c & ::= & \ldots \mid v \cdot e \end{array}$$

Une telle construction suggère l'idée de « Modus Tollens » : une preuve de $A \to B$, c'est un objet qui à partir d'un contexte d'évaluation de type B (vu comme une continuation de type $B^\perp$) construit un contexte d'évaluation de type A (vu comme une continuation de type $A^\perp$).

La règle de réduction associée est alors :

$$(\to^\perp) \quad \langle \lambda\beta.e \| v \cdot e' \rangle \quad \to \quad \langle \mu\beta.\langle \overline{v \| e} \rangle \| e' \rangle$$

Les règles de typage sont :

$$\frac{\Gamma \mid e : A \vdash \alpha : B, \Delta}{\Gamma \vdash \lambda\alpha.e : A \to B \,;\, \Delta} \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \,;\, v \cdot e : A \to B \vdash \Delta}$$

3.5 Produits

3.5.1 Produits à constructeurs additifs et multiplicatif

Le produit est associé du côté terme à la **paire** et du côté contexte d'évaluation à un contexte de décomposition de la paire.

La paire peut être une paire de valeurs (V_1, V_2) ou plus généralement une paire de termes arbitraires (v_1, v_2). Le contexte de décomposition de la paire peut être soit additif, c'est-à-dire construit à partir de **projections** (comme le connecteur additif « avec » de la logique linéaire), soit multiplicatif, c'est-à-dire construit à partir d'un lieur des composantes de la paire (comme le connecteur multiplicatif « tenseur » de la logique linéaire), appelé aussi `let` **destructurant**, ou **déstructuration liante de la paire**.

L'analyse du comportement calculatoire de LK via la logique linéaire a été explorée par Danos, Joinet et Schellinx [DJS97]. On trouvera aussi d'excellents commentaires dans les notes de Selinger sur la dualité catégorique entre CBN et CBV (sections 2 et 3 de [Sel03]). Ces références nous ont servi de repère.

On notera $\pi_1[e]$ et $\pi_2[e]$ les constructeurs de contexte d'évaluation additifs. Ces contextes prennent respectivement la première ou la deuxième composante de la paire avec laquelle ils interagissent puis continuent l'interaction avec e. Informellement, les règles de réduction associées sont donc

$$\begin{array}{lll} (\wedge_a^1) & \langle (v_1, v_2) \| \pi_1[e] \rangle \xrightarrow{h} \langle v_1 \| e \rangle \\ (\wedge_a^2) & \langle (v_1, v_2) \| \pi_2[e] \rangle \xrightarrow{h} \langle v_2 \| e \rangle \end{array}$$

On notera $(x, y).c$ (avec la contrainte $x \neq y$) le constructeur de contexte d'évaluation multiplicatif. C'est une forme de « let » destructurant à la ML. On se donne deux règles de réduction : ce contexte interagit avec une paire dont il lie les deux composantes aux variables x et y avant de continuer le calcul avec c. Informellement, il y a deux règles de réduction associées. Ces règles sont individuellement asymétriques mais symétriques l'une de l'autre.

$$\begin{array}{lll} (\wedge_m^1) & \langle (v_1, v_2) \| (x_1, x_2).c \rangle & \xrightarrow{h} \langle v_1 \| \tilde{\mu} x_1.\overline{\langle v_2 \| \tilde{\mu} x_2.c \rangle} \rangle \\ (\wedge_m^2) & \langle (v_1, v_2) \| (x_1, x_2).c \rangle & \xrightarrow{h} \langle v_2 \| \tilde{\mu} x_2.\overline{\langle v_1 \| \tilde{\mu} x_1.c \rangle} \rangle \end{array}$$

On laissera de côté le cas du calcul non-déterministe (qui est de toutes façons non confluent) et on ne s'intéressera qu'aux relations qu'entretient le produit avec les restrictions CBN et CBV.

Une première observation est que $(\wedge_m^1)$ et $(\wedge_m^2)$ forment une paire critique qui est convergente en CBN (car les variables x_1 et x_2 sont distinctes) mais pas en CBV. De fait, en CBN, on a

$$\langle v_1 \| \tilde{\mu} x_1.\overline{\langle v_2 \| \tilde{\mu} x_2.c \rangle} \rangle \rightarrow_{\tilde{\mu}} c[x_1 \leftarrow v_1][x_2 \leftarrow v_2] \leftarrow_{\tilde{\mu}} \langle v_2 \| \tilde{\mu} x_2.\overline{\langle v_1 \| \tilde{\mu} x_1.c \rangle} \rangle$$

tandis que pour $v_1 \triangleq \mu\_.\langle x_1 \| \alpha_1 \rangle$ et $v_2 \triangleq \mu\_.\langle x_2 \| \alpha_2 \rangle$ on a en CBV

$$\begin{array}{l} \langle v_1 \| \tilde{\mu} x_1.\overline{\langle v_2 \| \tilde{\mu} x_2.c \rangle} \rangle \rightarrow_{\mu} \langle x_1 \| \alpha_1 \rangle \\ \langle v_2 \| \tilde{\mu} x_2.\overline{\langle v_1 \| \tilde{\mu} x_1.c \rangle} \rangle \rightarrow_{\mu} \langle x_2 \| \alpha_2 \rangle \end{array}$$

Une seconde observation est que les contextes additifs et multiplicatifs sont interdéfinissables en CBN. En effet, quelque soit e, et quelque soit c dépendant de x_1 et x_2 on a

$$\begin{array}{l} \langle (v_1, v_2) \| (x_1, \_).\overline{\langle x_1 \| e \rangle} \rangle \xrightarrow{h}_{\wedge_m^1 \tilde{\mu}} \langle v_1 \| e \rangle \\ \langle (v_1, v_2) \| (\_, x_2).\overline{\langle x_2 \| e \rangle} \rangle \xrightarrow{h}_{\wedge_m^1 \tilde{\mu}} \langle v_2 \| e \rangle \\ \langle (v_1, v_2) \| \tilde{\mu} z.\overline{\langle z \| \pi_1[\tilde{\mu} x_1.\overline{\langle z \| \pi_2[\tilde{\mu} x_2.c] \rangle}] \rangle} \rangle \xrightarrow{h}_{\wedge_a^1 \wedge_a^2 \tilde{\mu}} \langle v_1 \| \tilde{\mu} x_1.\overline{\langle v_2 \| \tilde{\mu} x_2.c \rangle} \rangle \\ \langle (v_1, v_2) \| \tilde{\mu} z.\overline{\langle z \| \pi_2[\tilde{\mu} x_2.\overline{\langle z \| \pi_1[\tilde{\mu} x_1.c] \rangle}] \rangle} \rangle \xrightarrow{h}_{\wedge_a^1 \wedge_a^2 \tilde{\mu}} \langle v_2 \| \tilde{\mu} x_2.\overline{\langle v_1 \| \tilde{\mu} x_1.c \rangle} \rangle \end{array}$$

si bien que les contextes d'évaluation additifs sont définissables à partir du contexte d'évaluation multiplicatif par

$$\begin{array}{lll} \pi_1[e] & \triangleq & (x_1, \_).\langle x_1 \| e \rangle \\ \pi_2[e] & \triangleq & (\_, x_2).\langle x_2 \| e \rangle \end{array}$$

et, inversement, le contexte d'évaluation multiplicatif du produit peut être défini à partir des contextes additifs au choix par

$$\begin{array}{lll} (x_1, x_2).c & \triangleq & \tilde{\mu} z.\langle z \| \pi_2[\tilde{\mu} x_2.\overline{\langle z \| \pi_1[\tilde{\mu} x_1.c] \rangle}] \rangle \\ (x_1, x_2).c & \triangleq & \tilde{\mu} z.\langle z \| \pi_1[\tilde{\mu} x_1.\overline{\langle z \| \pi_2[\tilde{\mu} x_2.c] \rangle}] \rangle \end{array}$$

Le tout s'étend aux interactions avec les termes qui ne sont pas des valeurs.

Une troisième observation est que cette simulation peut dégrader la complexité en CBN, puisqu'une interaction telle que

$$\begin{array}{lll} \langle \mu\alpha.c' \| \tilde{\mu} z.\overline{\langle z \| \pi_1[\tilde{\mu} x_1.\overline{\langle z \| \pi_2[\tilde{\mu} x_2.c] \rangle}] \rangle} \rangle & \rightarrow_n & \langle \mu\alpha.c' \| \pi_1[\tilde{\mu} x_1.\overline{\langle \mu\alpha.c' \| \pi_2[\tilde{\mu} x_2.c] \rangle}] \rangle \\ & \rightarrow_n & c'[\alpha \leftarrow \pi_1[\tilde{\mu} x_1.\langle \mu\alpha.c' \| \pi_2[\tilde{\mu} x_2.c] \rangle]] \\ & \xrightarrow{*}_n & c'[\alpha \leftarrow \pi_1[\tilde{\mu} x_1.c'[\alpha \leftarrow \pi_2[\tilde{\mu} x_2.c]]]] \end{array}$$

peut dupliquer l'expression $\pi_2[\tilde{\mu} x_2.c]$ et les rédex que cette dernière expression peut créer au carré du nombre d'occurrences de α.

La quatrième observation est que les contextes d'évaluation additifs sont simulés en CBV par les contextes multiplicatifs seulement dans les interactions avec des paires de valeurs. Sinon, les évaluations divergent. Par exemple,

$$\begin{array}{lll} \langle (V, \mu\_.\overline{\langle x_0 \| v_0 \rangle}) \| (x_1, \_).\overline{\langle x_1 \| e \rangle} \rangle & \xrightarrow{*}_v & \langle x_0 \| v_0 \rangle \quad \text{tant pour } \wedge_m^1 \text{ que pour } \wedge_m^2 \\ \langle (V, \mu\_.\overline{\langle x_0 \| v_0 \rangle}) \| \pi_1[e] \rangle & \rightarrow_v & \langle V \| e \rangle \end{array}$$

Toutefois, les contextes multiplicatifs restent définissables à partir des contextes additifs.

Tout ceci suggère de ne considérer en CBV que la forme additive du produit. Quant au CBN, les deux formes étant mutuellement définissables, autant considérer la forme additive par similarité avec le CBV, à moins de souhaiter aussi la forme multiplicative pour des soucis de complexité algorithmique, mais alors présentée avec la règle de réduction

$$(\wedge_n^m) \qquad \langle (v_1, v_2) \| (x_1, x_2).c \rangle \xrightarrow{h} c[x_1 \leftarrow v_1][x_2 \leftarrow v_2]$$

afin d'éviter de briser arbitrairement la symétrie des règles.

En résumé, on définira donc le produit par les constructions suivantes :

$$\begin{array}{lll} V_c & ::= & \ldots \mid (v, v) \\ E_c & ::= & \ldots \mid \pi_1[e] \mid \pi_2[e] \end{array}$$

et les règles de réduction suivantes :

$$\begin{array}{llll} (\wedge_a^1) & \langle (v_1, v_2) \| \pi_1[e] \rangle & \xrightarrow{h} & \langle v_1 \| e \rangle \\ (\wedge_a^2) & \langle (v_1, v_2) \| \pi_2[e] \rangle & \xrightarrow{h} & \langle v_2 \| e \rangle \end{array}$$

Notons que la forme additive des contextes permet une forme de réduction faible des paires en appel par valeur. En effet, $\langle (v_1, v_2) \| \pi_1[e] \rangle$ se réduira en $\langle v_1 \| e \rangle$ sans avoir à évaluer v_2, alors que l'usage dans les langages de programmation en appel par valeur est d'avoir une réduction forte de la paire, c'est-à-dire une réduction des deux composantes même si l'une des composantes (voire les deux d'entre elles) est inutile pour la suite du calcul. Bien sûr, la paire forte de v_1 et v_2, forcément arbitrairement asymétrique, peut se définir via une $\tilde{\mu}$-expansion telle que $\mu\alpha.\langle v_1 \| \tilde{\mu} x_1.\overline{\langle v_2 \| \tilde{\mu} x_2.\overline{\langle (x_1, x_2) \| \alpha \rangle} \rangle} \rangle$.

3.5.2 Produit à constructeur multiplicatif asymétrique

Une autre approche pour les constructeurs de contexte multiplicatifs du produit est de leur associer d'emblée une syntaxe asymétrique. On a ainsi un destructeur privilégiant l'évaluation de la composante de gauche dont la syntaxe peut être notée :

$$E_c \quad ::= \quad \ldots \mid \lambda_1 x.e$$

et dont la règle de réduction associée est :

$$(\wedge_{m_1}) \qquad \langle (v_1, v_2) \| \lambda_1 x_1.e \rangle \xrightarrow{h} \langle v_1 \| \tilde{\mu} x_1.\overline{\langle v_2 \| e \rangle} \rangle$$

et un destructeur, symétrique du précédent, privilégiant l'évaluation de la composante de droite dont la syntaxe peut être notée :

$$E_c \quad ::= \quad \ldots \mid \lambda_2 x.e$$

et dont la règle de réduction associée est :

$$(\wedge_{m_2}) \qquad \langle (v_1, v_2) \| \lambda_2 x_2.e \rangle \xrightarrow{h} \langle v_2 \| \tilde{\mu} x_2.\overline{\langle v_1 \| e \rangle} \rangle$$

En particulier, $\lambda_1 x_1.\tilde{\mu} x_2.c$ équipé avec $(\wedge_{m_1})$ est équivalent à $(x_1, x_2).c$ équipé avec la règle $(\wedge_m^1)$, et symétriquement pour $\lambda_2 x_2.\tilde{\mu} x_1.c$ avec $(\wedge_{m_2})$ et $(\wedge_m^2)$.

En CBN, $\lambda_1 x_1.\tilde{\mu} x_2.c$ et $\lambda_2 x_2.\tilde{\mu} x_1.c$ se comportent de la même manière, l'utilité de $\lambda_1 x.e$ et $\lambda_2 x.e$ se résumant à des allègements de syntaxe.

En CBV, $\lambda_1 x_1.\tilde{\mu} x_2.c$ et $\lambda_2 x_2.\tilde{\mu} x_1.c$ se comportent différemment mais l'asymétrie de la syntaxe est alors en accord avec l'asymétrie de la réduction.

3.5.3 Produit et η-conversion

Trois règles d'η-conversion pour le produit additif sont susceptibles d'être étudiées :

$$
\begin{array}{llll}
\eta^R_{\wedge_a} & x & = & (\mu\alpha.\langle x\|\pi_1[\alpha]\rangle, \mu\beta.\langle x\|\pi_2[\beta]\rangle) \\
\eta^L_{\wedge^1_a} & \alpha & = & \tilde{\mu}z.\langle z\|\pi_1[\tilde{\mu}x.\overline{\langle z\|\pi_2[\tilde{\mu}y.\langle\overline{(x,y)\|\alpha}\rangle]\rangle}]\rangle \\
\eta^L_{\wedge^2_a} & \alpha & = & \tilde{\mu}z.\langle z\|\pi_2[\tilde{\mu}y.\overline{\langle z\|\pi_1[\tilde{\mu}x.\langle\overline{(x,y)\|\alpha}\rangle]\rangle}]\rangle
\end{array}
$$

La question se pose de déterminer s'il est justifié de qualifier les règles $\eta^L_{\wedge^1_a}$ et $\eta^L_{\wedge^2_a}$ de règles d'η-conversion. Par exemple, Danos, Joinet et Schellinx [DJS97] semblent appeler ces règles θ.

Les restrictions spécifiques des calculs en appel par nom et en appel par valeur impliquées par ces règles restent à être étudiées.

Par exemple, on conjecture que la restriction à l'appel par valeur ne construit des paires que de valeurs.

3.5.4 Typage des produits

L'introduction de la paire est typable comme suit

$$
\frac{\Gamma \vdash v_1 : A_1 \mid \Delta \qquad \Gamma \vdash v_2 : A_2 \mid \Delta}{\Gamma \vdash (v_1, v_2) : A_1 \wedge A_2 \,;\, \Delta}
$$

et l'introduction des constructeurs de contexte additifs du produit comme suit

$$
\frac{\Gamma \mid e : A_1 \vdash \Delta}{\Gamma \,;\, \pi_1[e] : A_1 \wedge A_2 \vdash \Delta} \qquad \frac{\Gamma \mid e : A_2 \vdash \Delta}{\Gamma \,;\, \pi_2[e] : A_1 \wedge A_2 \vdash \Delta}
$$

Les formes multiplicatives symétrique et asymétriques du contexte d'évaluation lié au produit sont quand à elles typables par

$$
\frac{c : (\Gamma, x_1 : A_1, x_2 : A_2 \vdash \Delta)}{\Gamma \,;\, (x_1, x_2).c : A_1 \wedge A_2 \vdash \Delta}
$$

$$
\frac{\Gamma, x : A_1 \mid e : A_2 \vdash \Delta}{\Gamma \,;\, \lambda_1 x.e : A_1 \wedge A_2 \vdash \Delta} \qquad \frac{\Gamma, x : A_2 \mid e : A_1 \vdash \Delta}{\Gamma \,;\, \lambda_2 x.e : A_1 \wedge A_2 \vdash \Delta}
$$

Il n'y a pas de raison a priori d'utiliser des connecteurs différents pour les produits additifs et multiplicatifs : les constructeurs de paire sont les mêmes et rien ne semble empêcher de laisser les deux types de constructeurs de contextes d'évaluation des paires coexister en CBN (voire en CBV si les paires sont restreintes à des paires de valeur).

3.6 Sommes

3.6.1 Sommes à constructeurs additifs et multiplicatif

Venons-en aux sommes. Par dualité, on obtient les sommes à constructeurs additifs avec la syntaxe

$$
\begin{array}{lll}
V_c & ::= & \ldots \mid \iota_1(v) \mid \iota_2(v) \\
E_c & ::= & \ldots \mid [e, e]
\end{array}
$$

Les expressions $\iota_1(v)$ et $\iota_2(v)$ représentent des **injections** tandis que l'expression $[e_1, e_2]$ est un contexte d'évaluation qui effectue une **analyse de cas** sur son argument, et, selon que cet argument est construit avec l'une ou l'autre des injections, applique l'argument de l'injection au contexte correspondant.

Les règles de réduction sont

$$
\begin{array}{llll}
(\vee^1_a) & \langle \iota_1(v)\|[E_1, E_2]\rangle & \xrightarrow{h} & \langle v\|E_1\rangle \\
(\vee^2_a) & \langle \iota_2(v)\|[E_1, E_2]\rangle & \xrightarrow{h} & \langle v\|E_2\rangle
\end{array}
$$

Pour les sommes à constructeurs multiplicatifs, la syntaxe est

$$\begin{array}{rcl} V_c & ::= & \ldots \mid [\alpha_1, \alpha_2].c \\ E_c & ::= & \ldots \mid [e, e] \end{array}$$

où $[\alpha_1, \alpha_2].c$ (avec la contrainte que les noms α_1 et α_2 sont distincts) est une forme de λ-abstraction qui lie deux contextes d'évaluation (c'est une implantation de la règle d'introduction droite du connecteur $⅋$ de la logique linéaire). Les règles de réduction sont :

$$\begin{array}{lrcl} (\vee_m^1) & \langle [\alpha_1, \alpha_2].c \| [e_1, e_2] \rangle & \xrightarrow{h} & \langle \mu\alpha_1.\langle \overline{\mu\alpha_2.c \| e_2} \rangle \| e_1 \rangle \\ (\vee_m^2) & \langle [\alpha_1, \alpha_2].c \| [e_1, e_2] \rangle & \xrightarrow{h} & \langle \mu\alpha_2.\langle \overline{\mu\alpha_1.c \| e_1} \rangle \| e_2 \rangle \end{array}$$

Ces règles introduisent une paire critique qui est convergente en CBV (car les noms α_1 et α_2 sont par définition distincts) mais pas en CBN.

Comme pour le produit, la somme à constructeurs additifs est plus primitive que la somme à constructeur multiplicatif dans la mesure où la seconde est simulable avec ses règles de réduction à partir des premières.

De plus, la compatibilité de la somme à constructeur multiplicatif avec le fragment intuitionniste est problématique du fait que le constructeur multiplicatif lie deux variables de contexte distinctes.

Les constructeurs additifs s'intègrent eux directement au fragment intuitionniste.

3.6.2 Somme à constructeur multiplicatif asymétrique

Comme dans le cas du produit, une autre approche pour les constructeurs de valeur multiplicatifs de la somme est de leur associer d'emblée une syntaxe asymétrique. On a ainsi un constructeur privilégiant l'évaluation de la composante de gauche dont la syntaxe peut être notée :

$$V_c \quad ::= \quad \ldots \mid \lambda_1\alpha.v$$

et dont la règle de réduction associée est :

$$(\vee_{m_1}) \qquad \langle \lambda_1\alpha_1.v \| [e_1, e_2] \rangle \quad \xrightarrow{h} \quad \langle \mu\alpha_1.\langle \overline{v \| e_2} \rangle \| e_1 \rangle$$

et un destructeur symétrique du précédent privilégiant l'évaluation de la composante de droite dont la syntaxe peut être notée :

$$V_c \quad ::= \quad \ldots \mid \lambda_2\alpha.e$$

et dont la règle de réduction associée est :

$$(\vee_{m_2}) \qquad \langle \lambda_2\alpha_2.v \| [e_1, e_2] \rangle \quad \xrightarrow{h} \quad \langle \mu\alpha_2.\langle \overline{v \| e_1} \rangle \| e_2 \rangle$$

En particulier, $\lambda_1\alpha_1.\mu\alpha_2.c$ équipé avec $(\vee_{m_1})$ est équivalent à $[\alpha_1, \alpha_2].c$ équipé avec la règle $(\vee_m^2)$, et symétriquement pour $\lambda_2\alpha_2.\mu\alpha_1.c$ avec $(\vee_{m_2})$ et $(\vee_m^2)$.

En CBV, $\lambda_1\alpha_1.\mu\alpha_2.c$ et $\lambda_2\alpha_2.\mu\alpha_1.c$ se comportent de la même manière, l'utilité de $\lambda_1\alpha..v$ et $\lambda_2\alpha..v$ se résumant à des allègements de syntaxe.

En CBN, $\lambda_1\alpha_1.\mu\alpha_2.c$ et $\lambda_2\alpha_2.\mu\alpha_1.c$ se comportent différemment mais l'asymétrie de la syntaxe est alors en accord avec l'asymétrie de la réduction.

Notons qu'un tel constructeur asymétrique de la somme a été considéré par Pym et Ritter lors de leur étude de la sémantique du calcul $\lambda\mu$ avec disjonction [PR01]. Ce constructeur asymétrique sera aussi utile lors de l'interprétation du système $\mu_n\tilde{\mu}$ avec lieurs dynamiques de contextes d'évaluation dans un langage sans lieurs dynamiques (voir section 4.4).

3.6.3 Typage de la somme

Les constructeurs additifs de la somme sont typable par

$$\frac{\Gamma \vdash v : A_1 \mid \Delta}{\Gamma \vdash \iota_1(v) : A_1 \vee A_2 \,;\, \Delta} \qquad \frac{\Gamma \vdash v : A_2 \mid \Delta}{\Gamma \vdash \iota_2(v) : A_1 \vee A_2 \,;\, \Delta}$$

tandis que le constructeur de contexte de la somme est typable par

$$\frac{\Gamma \,|\, e_1 : A_1 \vdash \Delta \qquad \Gamma \,|\, e_2 : A_2 \vdash \Delta}{\Gamma\,;\; [e_1, e_2] : A_1 \vee A_2 \vdash \Delta}$$

Les formes multiplicatives symétrique et asymétriques du terme lié à la somme sont quant à elles typables par

$$\frac{c : (\Gamma \vdash \alpha_1 : A_1, \alpha_2 : A_2, \Delta)}{\Gamma \vdash [\alpha_1, \alpha_2].c : A_1 \vee A_2 \,|\, \Delta}$$

$$\frac{\Gamma \vdash v : A_1 \,|\, \alpha : A_2, \Delta}{\Gamma \vdash \lambda_2 \alpha.v : A_1 \vee A_2\,;\, \Delta} \qquad \frac{\Gamma \vdash v : A_2 \,|\, \alpha : A_1, \Delta}{\Gamma \vdash \lambda_1 \alpha.v : A_1 \vee A_2\,;\, \Delta}$$

3.7 Constructeurs additifs de l'implication et de la soustraction

L'utilisation de constructeurs additifs peut aussi s'appliquer aux cas de l'implication et de la soustraction. Pour l'implication, la construction de contexte, qui est une construction binaire, reste inchangée. Ce qui est nouveau est la construction additive de terme qui est maintenant basée sur des injections :

$$\begin{array}{lll} V_c & ::= & \ldots \,\|\, \iota_1(e) \,\|\, \iota_2(v) \\ E_c & ::= & \ldots \,\|\, v \cdot e \end{array}$$

Les règles associées sont sans surprise :

$$\begin{array}{llll} (\rightarrow_a^1) & \langle \iota_1(e_1) \| v_1 \cdot e_2 \rangle & \xrightarrow{h} & \langle v_1 \| e_1 \rangle \\ (\rightarrow_a^2) & \langle \iota_2(v_2) \| v_1 \cdot e_2 \rangle & \xrightarrow{h} & \langle v_2 \| e_2 \rangle \end{array}$$

Les règles de typage des nouvelles constructions sont :

$$\frac{\Gamma \vdash v : B \,|\, \Delta}{\Gamma \vdash \iota_2(v) : A \rightarrow B \,|\, \Delta} \qquad \frac{\Gamma \,|\, e : A \vdash \Delta}{\Gamma \vdash \iota_1(e) : A \rightarrow B \,|\, \Delta}$$

De même, on peut utiliser un constructeur additif de contexte d'évaluation pour la soustraction :

$$\begin{array}{lll} V_c & ::= & \ldots \,\|\, e \cdot v \\ E_c & ::= & \ldots \,\|\, \pi_1[v] \,\|\, \pi_2[e] \end{array}$$

avec les règles associées :

$$\begin{array}{llll} (-_a^1) & \langle v_2 - e_1 \| \pi_1[e_1] \rangle & \xrightarrow{h} & \langle v_1 \| e_1 \rangle \\ (-_a^2) & \langle v_2 - e_1 \| \pi_2[v_2] \rangle & \xrightarrow{h} & \langle v_2 \| e_2 \rangle \end{array}$$

et les règles de typage :

$$\frac{\Gamma \,|\, e : B \vdash \Delta}{\Gamma \,|\, \pi_1[e] : B - A \vdash \Delta} \qquad \frac{\Gamma \vdash v : A \,|\, \Delta}{\Gamma \,|\, \pi_2[v] : B - A \vdash \Delta}$$

Notons que rien n'empêche de combiner les constructeurs additifs de l'implication et de la soustraction avec les constructeurs multiplicatifs.

Comme pour le produit et la somme, les constructeurs multiplicatifs sont dérivables des constructeurs additifs en posant :

$$\begin{array}{lll} \lambda x.v & \triangleq & \mu\alpha.\langle \iota_1(\tilde{\mu}x.\overline{\langle \iota_2(v) \| \alpha \rangle}) \| \alpha \rangle \\ \tilde{\lambda}\alpha.e & \triangleq & \tilde{\mu}x.\langle x \| \pi_1[\mu\alpha.\overline{\langle x \| \pi_2[e] \rangle}] \rangle \end{array}$$

En particulier, on vérifie que $(\rightarrow)$ et $(-)$ sont alors dérivables de $(\rightarrow_a^1)$, $(\rightarrow_a^2)$, $(-_a^1)$ et $(-_a^2)$.

Notons que la simulation de λ n'est pas valide dans la restriction intuitionniste puisqu'une liaison non linéaire de variable de contexte est nécessaire. D'où l'intérêt majeur de la λ-abstraction dans la conception d'un calcul compatible avec l'intuitionnisme.

Comme pour le produit et la somme, la définition des constructeurs additifs à partir des constructeurs multiplicatifs est soumise à restrictions. Plus précisément, la réduction en CBN n'est simulée que lors d'une interaction avec un contexte de la forme $v \cdot E$ et elle n'est simulée en CBV que lors d'une interaction avec un contexte de la forme $V \cdot e$.

Proposition 25 *Les constructeurs additifs de l'implication et leurs règles de réduction associées sont simulables dans les calculs* $\mu\tilde{\mu}_v^{\rightarrow^\eta}$ *et* $\mu_n\tilde{\mu}^{\rightarrow^\eta}$ *en posant*

$$\begin{array}{rcl} \iota_1(e) & \triangleq & \lambda x.\mu\_.\langle x \| e \rangle \\ \iota_2(v) & \triangleq & \lambda\_.v \end{array}$$

Notons que la simulation de ι_1 n'est pas valide dans la restriction intuitionniste puisqu'une liaison non linéaire de variable de contexte est nécessaire.

Par dualité, on a

Proposition 26 *Les constructeurs additifs de la soustraction et leurs règles de réduction associées sont simulables dans les calculs* $\mu\tilde{\mu}_v^{-^\eta}$ *et* $\mu_n\tilde{\mu}^{-^\eta}$ *en posant*

$$\begin{array}{rcl} \pi_1[v] & \triangleq & \tilde{\lambda}\alpha.\tilde{\mu}\_.\langle v \| \alpha \rangle \\ \pi_2[e] & \triangleq & \tilde{\lambda}\_.e \end{array}$$

Notons là que la définition de π_2 n'est pas valable dans le fragment intuitionniste.

3.8 Quantificateurs

Les quantificateurs universels et existentiels peuvent être interprétés non calculatoirement auquel cas ils sont respectivement interprétés au niveau du typage par des types intersection et union. Ils peuvent aussi être interprétés calculatoirement auquel cas ils sont respectivement interprétés par un produit dépendant et une somme dépendante.

On suppose donné un domaine de quantification dont les éléments sont notés par la lettre t et ses dérivés (ce domaine peut être un domaine d'individu de premier ordre, mais aussi de types, auquel cas la quantification est de second ordre). On utilise la lettre a pour désigner les variables parcourant le domaine de quantification. On réutilise pour le produit dépendant les notations de l'implication :

$$\begin{array}{rcl} V_c & ::= & \ldots \mid \lambda a.v \\ E_c & ::= & \ldots \mid t \cdot e \end{array}$$

La règle d'évaluation correspondante est

$$(\Pi) \qquad \langle \lambda a.v \| t \cdot e \rangle \xrightarrow{h} \langle v[a \leftarrow t] \| e \rangle$$

où $v[a \leftarrow t]$ désigne une substitution sans capture. La règle est compatible tant avec l'appel par nom que l'appel par valeur

La règle de typage correspondante est

$$\Pi_L \ \frac{\Gamma \mid e : A[a \leftarrow t] \vdash \Delta}{\Gamma\, ;\, t \cdot e : \Pi a.A \vdash \Delta} \qquad \Pi_R \ \frac{\Gamma \vdash v : A \mid \Delta}{\Gamma \vdash \lambda a.v : \Pi a.A\, ;\, \Delta}$$

avec la restriction que a n'apparaît pas libre dans Γ et Δ pour la règle Π_R.

La somme dépendante s'obtient par dualité avec la syntaxe

$$\begin{array}{rcl} V_c & ::= & \ldots \mid (t, v) \\ E_c & ::= & \ldots \mid \lambda a.e \end{array}$$

et la règle d'évaluation correspondante est

$$(\Sigma) \qquad \langle (t, v) \| \lambda a.e \rangle \quad \xrightarrow{h} \quad \langle v \| e[a \leftarrow t] \rangle$$

La règle de typage correspondante est

$$\Sigma_L \ \frac{\Gamma \,|\, e : A \vdash \Delta}{\Gamma \,;\, \lambda a.e : \Sigma a.A \vdash \Delta} \qquad \Sigma_R \ \frac{\Gamma \vdash v : A[a \leftarrow t] \,|\, \Delta}{\Gamma \vdash (t, v) : \Sigma a.A \,;\, \Delta}$$

avec la restriction que a n'apparaît pas libre dans Γ et Δ pour la règle Σ_L.

3.9 Vrai et faux

L'ajout de constructions correspondant aux connecteurs $\top$ (connecteur **vrai**) et $\perp$ (connecteur **faux**) est direct. Le connecteur $\top$ est lié à l'ajout d'une constant de terme $\ltimes$ que nous appelons **unité**.

$$V_c \quad ::= \quad \ldots \,\|\, \ltimes$$

Cette nouvelle constante n'a pas de contexte avec lequel interagir, aussi, elle ne vient avec aucune règle de réduction. En revanche, on peut lui associer une règle naturelle d'η-conversion qui s'avère assez violente dans un cadre non typé.

$$(\eta_\top) \quad \ltimes \quad = \quad x$$

Le typage de la construction associée au connecteur vrai se fait simplement avec la règle

$$\top_R \ \frac{}{\Gamma \vdash \ltimes : \top \,|\, \Delta}$$

La règle d'η-conversion affirme alors que l'unique habitant du type $\top$ est $\ltimes$.

Par dualité, le connecteur $\perp$ est lié à l'ajout d'une constante $\rtimes$ dans la catégorie des contextes d'évaluation.

$$E_c \quad ::= \quad \ldots \,\|\, \rtimes$$

Cette constante n'a pas de terme associé avec lequel interagir, aussi aucune règle de réduction n'est associée. On peut la typer avec

$$\perp_L \ \frac{}{\Gamma \,|\, \rtimes : \perp \vdash \Delta}$$

La règle associée d'η-conversion est

$$(\eta_\perp) \quad \rtimes \quad = \quad \alpha$$

3.10 Négation

La **négation** peut être construite via un constructeur unaire de valeur et un constructeur unaire de contexte d'évaluation linéaire, chacun permettant d'échanger la catégorie de l'expression concernée.

$$\begin{array}{lcl} V_c & ::= & \ldots \,\|\, \neg(e) \\ E_c & ::= & \ldots \,\|\, \neg[v] \end{array}$$

La règle de réduction naturelle associée est

$$(\neg) \qquad \langle \neg(e) \| \neg[v] \rangle \quad \xrightarrow{h} \quad \langle v \| e \rangle$$

Les règles de typage associées sont

$$\neg_L \ \frac{\Gamma \vdash v : A \,|\, \Delta}{\Gamma \,;\, \neg(v) : \neg A \vdash \Delta} \qquad \neg_R \ \frac{\Gamma \,|\, e : A \vdash \Delta}{\Gamma \vdash \neg[e] : \neg A \,;\, \Delta}$$

3.11 Des contextes d'évaluation récursifs aux fonctions récursives

Grâce à la soustraction, on peut coder n'importe quelle fonction récursive définie par une équation récursive de la forme :

$$f(x) \triangleq \psi(f, x)$$

En effet, grâce à la soustraction, on peut poser

$$f \triangleq \lambda x.\mu\alpha.\langle(\alpha, x)\|\tilde{\nu}_\beta.\psi'_\beta\rangle$$

où

$$\psi'_\beta \triangleq \tilde{\lambda}\gamma.\tilde{\mu}y.\langle\psi(\lambda x.\mu\alpha.\overline{\langle(\alpha, x)\|\beta\rangle}, y)\|\gamma\rangle$$

En effet, on a alors l'équation

$$\begin{array}{lll}\langle f\|v \cdot E\rangle & \to_n & \langle(E, v)\|\tilde{\nu}_\beta.\psi'_\beta\rangle \\ & \to_n & \langle(E, v)\|\tilde{\lambda}\gamma.\tilde{\mu}y.\overline{\langle\psi(\lambda x.\mu\alpha.\overline{\langle(\alpha, x)\|\tilde{\nu}_\beta.\psi'_\beta\rangle}, y)\|\gamma\rangle}\rangle \\ & \to_n & \langle\psi(\lambda x.\mu\alpha.\overline{\langle(\alpha, x)\|\tilde{\nu}_\beta.\psi'_\beta\rangle}, v)\|E\rangle \\ & = & \langle\psi(f, v)\|E\rangle\end{array}$$

Dans le cas où les appels à f dans ψ sont finals (« tail-recursive »), ψ' n'a pas besoin de propager une variable de continuation et on peut définir f plus simplement par

$$f \triangleq \lambda x.\mu\alpha.\langle x\|\tilde{\nu}_\beta.\psi'_{(\alpha,\beta)}\rangle$$

où

$$\psi'_{(\alpha,\beta)} \triangleq \tilde{\mu}y.\langle\psi(\lambda x.\mu\_.\overline{\langle x\|\beta\rangle}, y)\|\alpha\rangle$$

Notons aussi qu'une présentation duale des notions de récursion et corécursion apparaît dès Filinski [Fil89a, Fil89b].

3.12 Quelles notations pour le système $\mu\tilde{\mu}$ et ses extensions ?

Nous avons pour l'essentiel repris les notations développées dans [CH00]. L'expérience aidant, la barre centrale dans la notation de la coupure, en simple exemplaire dans [CH00], a ici été doublée, suivant la suggestion de Dougherty, Ghilezan et Lescanne [DGL04]. Cela permet de mieux séparer les deux composantes de la coupure.

Avec les notations de [CH00], il est difficile de décomposer une expression complexe du système $\mu\tilde{\mu}$ en ses différents éléments, et en particulier de déterminer l'étendue visuelle des composantes des coupures, d'autant plus que les coupures sont souvent imbriquées.

Pour pallier ce problème, nous avons adopté le principe de surligner les coupures sous-termes d'une autre coupure. De plus, ces coupures sous-termes sont imprimées en léger grisé pour les faire ressortir de la partie de l'expression la plus externe. Par ce biais, la structure arborescente des expressions est plus facilement saisie par l'œil.

À la réflexion, nous pensons qu'une amélioration notable de la lisibilité serait atteinte si l'on adoptait le principe que la coupure n'est pas nécessairement écrite dans l'ordre terme-contexte mais qu'elle peut l'être aussi dans l'ordre contexte-terme. En particulier, nous pensons que les coupures dont l'un des côtés est une variable devraient être écrites en commençant par la variable, qu'elle soit une variable de terme ou de contexte d'évaluation. À la manière des quantifications de la forme $\forall x \geq$ *longue-expression* ou $\forall x \leq$ *longue-expression* qui s'arrangent pour avoir la variable en premier et choisissent en conséquence le signe $\geq$ ou $\leq$, nous pensons que dans une coupure avec une variable et une expression qui n'est pas une variable, la pensée est d'abord pour la variable, et qu'il est pertinent de l'écrire d'abord. Comme, la plupart des expressions intéressantes sont précisément constituées de coupure dont l'un des côtés est une variable, nous pensons que ce genre d'heuristique permettrait d'écrire et de lire les expressions comme on les pense. Quitte à ce qu'il y ait une part d'arbitraire dans le cas de coupures entre deux variables.

Voici par exemple comment l'on écrirait la preuve du tiers-exclu :

$$\mu\alpha.\langle\alpha\|\iota_1(\lambda x.\mu\_.\overline{\langle\alpha\|\iota_2(x)\rangle})\rangle$$

On remarque en particulier que c'est une notation qui rapproche de la syntaxe du calcul $\lambda\mu$.

À côté des questions de représentation graphique de la coupure, on trouve les questions relatives à la symétrie du calcul. Les contrastes entre μ et $\tilde{\mu}$ et entre (v_1, v_2) et $[e_1, e_2]$ sont intéressants mais manquent d'uniformité.

Nous avons renoncé à l'opposition entre $\lambda x.v$ et $\beta\lambda.e$ de [CH00] en raison de la difficulté à lire $\beta\lambda$ (sans doute parce que le β n'est pas délimité par le λ et le point comme l'est le x dans $\lambda x.v$, mais peut-être aussi parce qu'on ne retrouve pas l'orientation de la barre principale du λ vers la variable liée, et que la lecture de gauche à droite se retrouve embarrassée avec un β dont le rôle ne se fait connaître qu'à la lecture du λ, ou peut-être est-ce simplement la force de l'usage du λx que l'on arrive pas à retrouver dans le $\beta\lambda$). Nous lui avons préféré la notation $\tilde{\lambda}\beta.e$ qui a l'avantage d'avoir la même structure nom-du-lieur/nom-de-la-variable/séparateur/objet-quantifié que le $\lambda x.v$, et de suivre le même principe de symétrie que μ et $\tilde{\mu}$.

Nous n'avons pas trouvé à l'heure actuelle de notations pour les injections et les projections qui soient à la fois symétriques et réminiscentes de l'usage.

Le constructeur de terme de la soustraction était noté $e \cdot v$ dans [CH00]. L'ordre entre e et v est tel que la confusion avec le contexte applicatif $v \cdot e$ est évitée, mais cela reste non complètement satisfaisant pour les raisons suivantes : 1) $e \cdot v$ est censé représenter « v moins e », on se serait attendu à ce que l'ordre soit v puis e, 2) le contexte contre lequel un tel terme est appliqué lie d'abord le contexte e, là encore on se serait attendu à ce que l'ordre soit inversé. C'est pourquoi nous avons plutôt utilisé ici la notation $v - e$. Reste qu'il n'y a pas de symétrie graphique entre le trait et le point de $v \cdot e$. Une autre notation pour $v - e$ aurait pu être $v \mathbin{\tilde{\cdot}} e$, avec la même opposition qu'entre λ et $\tilde{\lambda}$ et qu'entre μ et $\tilde{\mu}$, … sauf que cette fois, le tilde apparaît dans la catégorie des termes. Peut-être plus troublant qu'autre chose.

L'orientation gauche-droite de la lecture fait qu'une coupure $\langle \mu\alpha.\mathit{une\text{-}longue\text{-}expression} \| e\rangle$ est plus difficile à saisir que la coupure duale $\langle v \| \tilde{\mu}x.\mathit{une\text{-}longue\text{-}expression}\rangle$. La possibilité d'échanger l'ordre des composantes de la coupure pourrait parfois améliorer la situation (on écrirait par exemple plutôt $\langle e \| \mu\alpha.\mathit{une\text{-}longue\text{-}expression}\rangle$).

D'un autre côté, les notations adoptées par Wadler [Wad03] sont remarquablement symétriques. En particulier la notation $(c).\alpha$ pour $\mu\alpha.c$ est parfaitement symétrique de la notation $x.(c)$ pour $\tilde{\mu}x.c$. Un défaut peut-être est le manque de signes distinctifs, hors le point et la parenthèse associée, permettant de saisir d'emblée le rôle de x et de α dans l'expression. Mais cela peut être une question d'habitude.

3.13 Relation entre les calculs $\overline{\lambda}\mu\tilde{\mu}$ et $\lambda\mu$

La structure fine des calculs $\overline{\lambda}\mu\tilde{\mu}$ et $\lambda\mu$ diffère par la forme des applications (en $\overline{\lambda}\mu\tilde{\mu}$, la fonction est en tête de terme alors qu'elle est enfouie au fond de la structure applicative dans le calcul $\lambda\mu$). En particulier, les étapes élémentaires de propagation de la substitution dans les deux calculs ne sont pas isomorphes. Toutefois, on peut établir plusieurs relations entre les calculs $\overline{\lambda}\mu\tilde{\mu}$ et $\lambda\mu$.

Déjà, la coupure et le constructeur de contexte applicatif du calcul $\overline{\lambda}\mu\tilde{\mu}$ sont plus élémentaires que l'application du calcul $\lambda\mu$ dont ils fournissent précisément une décomposition fine. Ainsi, on peut définir des plongements compositionnels (c'est-à-dire invariant par passage au contexte) du calcul $\lambda\mu$ dans le calcul $\overline{\lambda}\mu\tilde{\mu}$. Deux plongements compositionnels canoniques seront étudiés en section 3.13.2. Chacun des plongements enverra le calcul $\lambda\mu$ par nom vers le calcul $\overline{\lambda}\mu\tilde{\mu}$ par nom, modulo un ensemble de règles de réduction linéaires mais le premier plongement enverra le calcul $\lambda\mu$ par valeur *gauche-droite* vers le calcul $\overline{\lambda}\mu\tilde{\mu}$ par valeur tandis que c'est le calcul $\lambda\mu$ par valeur *droite-gauche* que le second plongement enverra vers le calcul $\overline{\lambda}\mu\tilde{\mu}$ par valeur.

À côté des plongements compositionnels, on peut aussi s'intéresser à des plongements, qui sont non compositionnels mais qui préservent les formes normales. On peut en particulier définir de tels plongements, tant pour l'appel par nom et que pour l'appel par valeur. On verra en sections 3.13.3 et 3.13.4 à quelles conditions on peut faire de ces plongements des isomorphismes, ce qui donnera une idée des différences structurelles entre les calculs $\lambda\mu$ et $\overline{\lambda}\mu\tilde{\mu}$.

La question des connexions entre $\overline{\lambda}\mu\tilde{\mu}$ non-déterministe et $\lambda\mu$ non-déterministe reste ouverte.

3.13.1 Le calcul $\lambda\mu$

Nous rappelons la syntaxe du $\lambda\mu$-calcul de Parigot [Par92].

$$\begin{array}{lcl} V & ::= & x \mid \lambda x.v \\ v & ::= & V \mid v\, v \mid \mu\alpha.c \\ c & ::= & [\alpha]v \end{array}$$

Plusieurs systèmes de réduction peuvent être considérés. Nous mentionnons d'abord les réductions déterministes : l'appel par nom, originellement introduit par Parigot, l'appel par valeur avec évaluation des applications de gauche à droite tel qu'étudié par Ong et Stewart [OS97], l'appel par valeur avec évaluation des applications de droite à gauche tel que mentionné dans l'article fondateur de Parigot. Nous mentionnons ensuite les systèmes de réduction non-déterministe : le calcul non-déterministe CBN et CBV_{LR} et le non-déterministe CBN, CBV_{LR} et CBV_{RL} tel qu'étudié par David et Nour [DN05]. Ces deux calculs mériteraient d'être comparés au calcul $\overline{\lambda}\mu\tilde{\mu}$ non-déterministe.

Aux différents calculs, on associe aussi les règles d'η-réduction suivantes :

$$\begin{array}{lllll} (\eta_\mu) & \mu\alpha.[\alpha]v & \xrightarrow{h} & v & \alpha \text{ non libre dans } v \\ (\eta_v) & \lambda x.(V\, x) & \xrightarrow{h} & V & x \text{ non libre dans } V \\ (\eta) & \lambda x.(v\, x) & \xrightarrow{h} & v & x \text{ non libre dans } v \end{array}$$

où (η_μ) est valable pour tous les calculs, (η_V) est pour les calculs par valeur déterministes et (η) est pour le calcul par nom et les calculs non-déterministes.

Dans tous les cas, si E est un contexte d'évaluation défini par

$$E \quad ::= \quad \Box \mid v\, E \mid E\, v$$

alors la notation $c[\beta \leftarrow [\alpha]E]$ désigne la substitution (sans capture) des sous-termes de la forme $[\beta]v$ dans c par $[\alpha](E[v])$.

Appel par nom

$$\begin{array}{llll} (\beta) & (\lambda x.v)\, v' & \xrightarrow{h} & v[x \leftarrow v'] \\ (\mu_L) & (\mu\alpha.c)\, v' & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](\Box\, v')] \\ (\mu_{var}) & [\alpha]\mu\beta.c & \xrightarrow{h} & c[\beta \leftarrow [\alpha]\Box] \end{array}$$

Appel par valeur (gauche-droite)[3]

$$\begin{array}{llll} (\beta_v) & (\lambda x.v)\, V & \xrightarrow{h} & v[x \leftarrow V] \\ (\mu_L) & (\mu\alpha.c)\, v & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](\Box\, v)] \\ (\mu_{Rv}) & (\lambda x.v)\, (\mu\alpha.c) & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha]((\lambda x.v)\, \Box)] \\ (\mu_{var}) & [\alpha]\mu\beta.c & \xrightarrow{h} & c[\beta \leftarrow [\alpha]\Box] \end{array}$$

Appel par valeur (droite-gauche)

$$\begin{array}{llll} (\beta_v) & (\lambda x.v)\, V & \xrightarrow{h} & v[x \leftarrow V] \\ (\mu_{Lv}) & (\mu\alpha.c)\, V & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](\Box\, V)] \\ (\mu_R) & v\, (\mu\alpha.c) & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](v\, \Box)] \\ (\mu_{var}) & [\alpha]\mu\beta.c & \xrightarrow{h} & c[\beta \leftarrow [\alpha]\Box] \end{array}$$

Non-déterministe appel par nom et appel par valeur (gauche-droite)

[3] Les règles (μ_L) et (μ_{Rv}) sont respectivement appelées (ζ_{fun}) et (ζ_{arg}) dans Ong et Stewart [OS97]. La règle (ζ_{arg}) est en fait plus générale puisqu'elle s'applique aussi au cas où $\lambda x.v$ est remplacé par une variable. Toutefois, puisque de toutes façons, le calcul est « incomplet » vis à vis de la réduction des termes ouverts, nous avons écarté les instances manifestement non closes de la règle.

$$
\begin{array}{llcl}
(\beta) & (\lambda x.v)\, v' & \xrightarrow{h} & v[x \leftarrow v'] \\
(\mu_L) & (\mu\alpha.c)\, v & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](\Box\, v)] \\
(\mu_{Rv}) & (\lambda x.v)\,(\mu\alpha.c) & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha]((\lambda x.v)\,\Box)] \\
(\mu_{var}) & [\alpha]\mu\beta.c & \xrightarrow{h} & c[\beta \leftarrow [\alpha]\Box]
\end{array}
$$

Non-déterministe appel par nom et appels par valeur

$$
\begin{array}{llcl}
(\beta) & (\lambda x.v)\, v' & \xrightarrow{h} & v[x \leftarrow v'] \\
(\mu_L) & (\mu\alpha.c)\, v & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](\Box\, v)] \\
(\mu_R) & v\,(\mu\alpha.c) & \xrightarrow{h} & \mu\alpha.c[\alpha \leftarrow [\alpha](v\,\Box)] \\
(\mu_{var}) & [\alpha]\mu\beta.c & \xrightarrow{h} & c[\beta \leftarrow [\alpha]\Box]
\end{array}
$$

3.13.2 Plongements compositionnels du calcul $\lambda\mu$ dans le calcul $\overline{\lambda}\mu\tilde{\mu}$

Les deux plongements compositionnels suivants viennent de [CH00] :

$$
\begin{array}{lcll}
x^{>} & \triangleq & x & \\
(\lambda x.v)^{>} & \triangleq & \lambda x.v^{>} & \\
(v\ v')^{>} & \triangleq & \mu\alpha.\langle v^{>} \| v'^{>} \cdot \alpha\rangle & \text{pour } \alpha \text{ frais} \\
(\mu\alpha.c)^{>} & \triangleq & \mu\alpha.c^{>} & \\
([\alpha]v)^{>} & \triangleq & \langle v^{>} \| \alpha\rangle &
\end{array}
$$

$$
\begin{array}{lcll}
x^{<} & \triangleq & x & \\
(\lambda x.v)^{<} & \triangleq & \lambda x.v^{<} & \\
(v\ v')^{<} & \triangleq & \mu\alpha.\langle v'^{<} \| \tilde{\mu}x.\overline{\langle v^{<} \| x \cdot \alpha\rangle}\rangle & \text{pour } x \text{ et } \alpha \text{ frais} \\
(\mu\alpha.c)^{<} & \triangleq & \mu\alpha.c^{<} & \\
([\alpha]v)^{<} & \triangleq & \langle v^{<} \| \alpha\rangle &
\end{array}
$$

Le premier plongement (dont on peut remarquer que la restriction au λ-calcul est essentiellement le plongement de la déduction naturelle dans LJ de Gentzen [Gen35]) a la propriété de plonger les calculs $\lambda\mu$ par nom et par valeur gauche-droite vers les restrictions appel par nom et appel par valeur du calcul $\overline{\lambda}\mu\tilde{\mu}$. Avec le second plongement[4], c'est en revanche le calcul $\lambda\mu$ par valeur droite-gauche qui est envoyé vers la restriction appel par valeur du calcul $\overline{\lambda}\mu\tilde{\mu}$ (tandis que le calcul $\lambda\mu$ par nom reste envoyé vers le calcul $\overline{\lambda}\mu\tilde{\mu}$ par nom).

Aucun des deux plongements ne préservent les formes normales. Par exemple, la traduction de $x\,y\,z$ par $^{>}$ est $\mu\alpha.\langle\mu\alpha'.\overline{\langle x\|y\cdot\alpha'\rangle}\|z\cdot\alpha\rangle$ dont la forme normale est $\mu\alpha.\langle x\|y\cdot z\cdot\alpha\rangle$.

Sur la restriction intuitionniste, on vérifie la propriété suivante :

Proposition 27 *Le plongement $^{>}$ est un morphisme pour (β) (resp. (β_v)) du λ-calcul vers la restriction intuitionniste du calcul $\overline{\lambda}\mu\tilde{\mu}$ en appel par nom (resp. en appel par valeur)*

Preuve: On a

$$
\begin{array}{rcl}
((\lambda x.v)\, v')^{>} & = & \mu\alpha.\langle\lambda x.v^{>}\|v'^{>}\cdot\alpha\rangle \\
& \rightarrow_{\rightarrow} & \mu\alpha.\langle v'^{>}\|\tilde{\mu}x.\overline{\langle v^{>}\|\alpha\rangle}\rangle \\
& \rightarrow_{\mu} & \mu\alpha.\langle v^{>}[x \leftarrow v'^{>}]\|\alpha\rangle \\
& \rightarrow_{\eta_\mu} & v^{>}[x \leftarrow v'^{>}] \\
& = & v[x \leftarrow v']^{>}
\end{array}
$$

De même pour β_v. ■

Sur le fragment non intuitionniste, en revanche, la simulation n'est pas immédiate. On a

Proposition 28 *Si $t \rightarrow t'$ dans le calcul $\lambda\mu$ en appel par nom alors, il existe v tel que $t^{>} \rightarrow_n v \xleftarrow{*}_{\mu} t'^{>}$.*

[4]Sur l'isomorphisme entre les calculs $\lambda\mu$ par valeur gauche-droite et par valeur droite-gauche, on lira avec intérêt l'analyse de Selinger [Sel03].

PREUVE: On remarque d'abord que

$$\begin{array}{lll} c[\alpha \leftarrow [\alpha](\Box v)]^{>} & \xrightarrow{*}_{\mu} & c^{>}[\alpha \leftarrow v^{>} \cdot \alpha] \\ c[\alpha \leftarrow [\alpha]((\lambda x.v)\,\Box)]^{>} & \xrightarrow{*}_{\mu\rightarrow} & c^{>}[\alpha \leftarrow \tilde{\mu}x.\langle v^{>} \| \alpha\rangle] \end{array}$$

D'où

$$\begin{array}{lll} (\mu\alpha.c\ v)^{>} & = & \mu\alpha'.\langle \mu\alpha.c^{>} \| v^{>} \cdot \alpha'\rangle \\ & \rightarrow_{\mu} & \mu\alpha'.c^{>}[\alpha \leftarrow v^{>} \cdot \alpha'] \\ & \xleftarrow{*}_{\mu} & (\mu\alpha'.c[\alpha \leftarrow [\alpha](\Box v)])^{>} \end{array}$$

$$\begin{array}{lll} ((\lambda x.v)\ (\mu\alpha.c))^{>} & = & \mu\alpha'.\langle \lambda x.v^{>} \| \mu\alpha.c^{>} \cdot \alpha'\rangle \\ & \rightarrow_{\rightarrow} & \mu\alpha'.\langle \mu\alpha.c^{>} \| \tilde{\mu}x.\langle \overline{v^{>} \| \alpha'}\rangle\rangle \\ & \rightarrow_{\mu} & \mu\alpha'.c^{>}[\alpha \leftarrow \tilde{\mu}x.\langle v^{>} \| \alpha'\rangle] \\ & \xleftarrow{*}_{\mu\rightarrow} & (\mu\alpha'.c[\alpha \leftarrow [\alpha]((\lambda x.v)\ \Box)])^{>} \end{array}$$

Par ailleurs

$$\begin{array}{lll} ([\beta]\mu\gamma.c)^{>} & = & \langle \mu\gamma.c^{>} \| \beta\rangle \\ & \rightarrow & c^{>}[\beta \leftarrow \gamma] \\ & = & (c[\gamma \leftarrow [\beta]\Box])^{>} \end{array}$$

■

Quant à la simulation de l'appel par valeur gauche-droite, elle n'est pas très élégante en raison de la confusion entre `let-in` et β-rédex. Telles que les règles du calcul $\lambda\mu$ par valeur gauche-droite sont exprimées, on a :

Proposition 29 *Si $t \rightarrow t'$ dans le calcul $\lambda\mu$ en appel par valeur gauche-droite alors, il existe v tel que $t^{>} \rightarrow_{v} v \xleftarrow{*}_{\mu\rightarrow} t'^{>}$.*

PREUVE: Ceci est dû au fait que

$$c[\alpha \leftarrow [\alpha]((\lambda x.v)\,\Box)]^{>} \quad \xrightarrow{*}_{\mu\rightarrow} \quad c^{>}[\alpha \leftarrow \tilde{\mu}x.\langle v^{>} \| \alpha\rangle]$$

D'où

$$\begin{array}{lll} ((\lambda x.v)\ (\mu\alpha.c))^{>} & = & \mu\alpha'.\langle \lambda x.v^{>} \| \mu\alpha.c^{>} \cdot \alpha'\rangle \\ & \rightarrow_{\rightarrow} & \mu\alpha'.\langle \mu\alpha.c^{>} \| \tilde{\mu}x.\langle \overline{v^{>} \| \alpha'}\rangle\rangle \\ & \rightarrow_{\mu} & \mu\alpha'.c^{>}[\alpha \leftarrow \tilde{\mu}x.\langle v^{>} \| \alpha'\rangle] \\ & \xleftarrow{*}_{\mu\rightarrow} & (\mu\alpha'.c[\alpha \leftarrow [\alpha]((\lambda x.v)\ \Box)])^{>} \end{array}$$

■

Le plongement $^{<}$ a été étudié par Rocheteau [Roc05]. On peut reformuler ses résultats comme suit :

Lemme 3 (Rocheteau) *On a les correspondances suivantes, des réductions du calcul $\lambda\mu$ vers les réductions du calcul $\overline{\lambda}\mu\tilde{\mu}$:*

- *si $t \rightarrow_{\beta} t'$ alors $t^{<} \rightarrow_{\rightarrow\tilde{\mu}^{l}_{n}\eta_{\mu}} t'^{<}$*
- *si $t \rightarrow_{\beta_v} t'$ alors $t^{<} \rightarrow_{\rightarrow\tilde{\mu}^{l}\eta_{\mu}} t'^{<}$*
- *si $t \rightarrow_{\mu_L} t'$ alors il existe v tel que $t^{<} \rightarrow_{\tilde{\mu}^{l}_{n}\mu} v \xleftarrow{*}_{\mu^{l}\tilde{\mu}^{l}_{n}} t'^{<}$*
- *si $t \rightarrow_{\mu_{Lv}} t'$ alors il existe v tel que $t^{<} \rightarrow_{\tilde{\mu}^{l}\mu} v \xleftarrow{*}_{\mu^{l}\tilde{\mu}^{l}} t'^{<}$*
- *si $t \rightarrow_{\mu_R} t'$ alors il existe v tel que $t^{<} \rightarrow_{\mu_v} v \xleftarrow{*}_{\mu^{l}} t'^{<}$*
- *si $t \rightarrow_{\mu_{var}} t'$ alors $t^{<} \rightarrow_{\mu} t'^{<}$*

où μ^{l} et $\tilde{\mu}^{l}$ désignent des applications linéaires des règles μ et $\tilde{\mu}$ (c'est-à-dire des applications pour lesquelles la variable liée apparaît une fois et une seule).

Proposition 30 (Rocheteau)
Si $t \xrightarrow{*} t'$ dans le calcul $\lambda\mu$ par nom alors il existe v tel que $t^{<} \xrightarrow{*}_{n} v \xleftarrow{*}_{\mu^{l}\tilde{\mu}^{l}_{n}} t'^{<}$.
Si $t \xrightarrow{*} t'$ dans le calcul $\lambda\mu$ par valeur droite-gauche alors il existe v tel que $t^{<} \xrightarrow{*}_{v} v \xleftarrow{*}_{\mu^{l}\tilde{\mu}^{l}} t'^{<}$.

En fait, Rocheteau montre plus, à savoir que $^{<}$ constitue un isomorphisme modulo les réductions linéaires $=_{\mu^{l}\tilde{\mu}^{l}}$ entre

– d'une part le calcul $\lambda\mu$ par nom (avec les règles (β), (μ_L), (μ_{var}) et (η_μ)) et le calcul $\overline{\lambda}\mu\tilde{\mu}$ par nom (avec les règles $(\to^\beta)$, (μ_n), $(\tilde{\mu})$, (η_μ) et la restriction des contextes applicatifs à la forme $v \cdot E$)
– d'autre part le calcul $\lambda\mu$ par valeur droite-gauche (avec les règles (β_v), (μ_{Lv}), (μ_R), (μ_{var}) et (η_μ)) et le calcul $\overline{\lambda}\mu\tilde{\mu}$ par valeur (avec les règles $(\to_v^\beta)$, (μ), $(\tilde{\mu}_v)$, (η_μ) et la restriction des contextes applicatifs à la forme $V \cdot e$).

Il est probable qu'un tel isomorphisme pourrait être prouvé avec $^>$ pour l'appel par nom et pour l'appel par valeur *gauche-droite*.

3.13.3 Conditions d'un isomorphisme entre les calculs $\lambda\mu$ et $\overline{\lambda}\mu\tilde{\mu}$ en appel par nom

Le plongement $^{\mathsf{N}}$ suivant vient de [CH00] (où il est noté $^{\mathcal{N}}$) :

$$
\begin{array}{lcll}
x^{\mathsf{N}} & \triangleq & x & \\
(\lambda x.v)^{\mathsf{N}} & \triangleq & \lambda x.v^{\mathsf{N}} & \\
(\mu\alpha.c)^{\mathsf{N}} & \triangleq & \mu\alpha.c^{\mathsf{N}} & \\
(v\ v')^{\mathsf{N}} & \triangleq & \mu\alpha.(v\,v')^{\mathsf{N}}_\alpha & \text{pour } \alpha \text{ frais} \\
 & & & \\
([\alpha]v)^{\mathsf{N}} & \triangleq & v^{\mathsf{N}}_\alpha & \\
 & & & \\
(v\ v')^{\mathsf{N}}_e & \triangleq & v^{\mathsf{N}}_{v'^{\mathsf{N}}\cdot e} & \\
v^{\mathsf{N}}_e & \triangleq & \langle v \| e\rangle & \text{si } v \text{ n'est pas une application}
\end{array}
$$

Ce plongement étend au cas non typé le plongement des formes normales de la déduction naturelle classique (en appel par nom) vers le calcul des séquents par Prawitz[5] [Pra65]. Il explicite le fait que la cible de la traduction de Prawitz est le calcul des séquents en appel par nom.

La condition nécessaire du résultat suivant vient de [CH00].

Proposition 31 *v est normal dans le calcul $\lambda\mu$ en appel par nom ssi v^{N} est normal dans le calcul $\overline{\lambda}\mu\tilde{\mu}$.*

Par exemple, la traduction d'une expression normale de la forme

$$\lambda x_1 \ldots x_n.\mu\alpha.[\beta]y\, v_1 \ldots v_p$$

est

$$\lambda x_1 \ldots x_n.\mu\alpha.\langle y \| v_1 \cdot \ldots \cdot v_p \cdot \beta\rangle\,.$$

À certaines conditions, le plongement $^{\mathsf{N}}$ sera un isomorphisme entre les variantes CBN des calculs $\lambda\mu$ et $\overline{\lambda}\mu\tilde{\mu}$. Le prix pour obtenir un isomorphisme est de restreindre le calcul $\lambda\mu$ à une sous-classe de termes expansés vis à vis de la règle (η_μ) et de décomposer la règle de réduction (μ) du calcul $\overline{\lambda}\mu\tilde{\mu}$. De fait, la réduction du calcul $\lambda\mu$ substitue les contextes de manière incrémentale. Pour espérer une simulation, il faut transposer ce principe de substitution incrémentale des contextes d'évaluation au cas du $\overline{\lambda}\mu\tilde{\mu}$. Remplaçons donc la règle (μ) du $\overline{\lambda}\mu\tilde{\mu}$ par les règles suivantes

$$
\begin{array}{lrcl}
(\mu_{app}) & \langle \mu\beta.c \| v \cdot E\rangle & \to & \langle \mu\beta.(c[\beta \leftarrow v \cdot \beta]) \| E\rangle \\
(\mu_{var}) & \langle \mu\beta.c \| \alpha\rangle & \to & c[\beta \leftarrow \alpha]
\end{array}
$$

On appelle réduction μ-atomique le remplacement de la réduction (μ) par la décomposition ci-dessous.

La traduction $^{\mathsf{N}}$ envoie différents termes du calcul $\lambda\mu$ vers le même terme du calcul $\overline{\lambda}\mu\tilde{\mu}$. Par exemple $\lambda x.(x\,y)$ et $\lambda x.\mu\alpha.[\alpha](x\,y)$ ont la même image. Les deux expressions diffèrent par une η_μ-conversion :

$$\mu\alpha.[\alpha]v \quad = \quad v \qquad \text{si } \alpha \text{ n'apparaît pas dans } c$$

[5]La transformation de Prawitz, présentée pour un calcul des séquents dont les contextes de formules sont des ensembles de formules est hypothético-amalgamante (cf page 1.2) même si la déduction naturelle source n'est pas, elle, définie de manière hypothético-amalgamante. De plus, la transformation telle que présentée est incorrecte dans le cas classique pour les séquents mentionnant la formule $\perp$ en position arbitraire (l'hypothèse d'induction qui sépare les hypothèses de la déduction naturelle en hypothèses ou conclusions du calcul des séquents n'est pas compatible avec l'hypothèse d'induction attendue lors de l'étape de traduction de Modus Ponens appliqué à une formule de la forme $(C_1 \to \perp) \to C_2$). Avec cette restriction sur les occurrences de $\perp$, les dérivations de $\perp$ en déduction naturelle s'interprètent comme des dérivations de séquents sans formule non nommée (au sens de Parigot), c'est-à-dire que $\perp$ joue le rôle de $\perp\!\!\!\perp$ au sens du calcul $\lambda_{\mathcal{C}-\mathtt{tp}}$ de Ariola *et al* [AH03, AHS05a, AFH05].

Pour faire de $^{\mathsf{N}}$ une injection, on restreint les termes du calcul $\lambda\mu$ à certains représentants de classes d'η_μ-équivalence. On dit qu'un terme du calcul $\lambda\mu$ est η_μ^{app}-expansé si on ne peut remplacer aucun sous-terme applicatif v par $\mu\alpha.[\alpha]v$ (pour α frais) sans créer de rédex. Par exemple, $\lambda x.\mu\alpha.[\alpha](x\,\mu\beta.[\beta](y\,z))$, $\lambda x.x$ et $\mu\alpha.[\alpha](x\,y)$ sont η_μ^{app}-expansés mais pas $\lambda x.(x\,(y\,z))$ ni $(x\,y)$.

L'image du calcul $\lambda\mu$ par $^{\mathsf{N}}$ ne contient pas de termes contenant un $\tilde{\mu}$. Pour faire de $^{\mathsf{N}}$ une surjection, on se restreint donc à la restriction sans $\tilde{\mu}$ du calcul $\overline{\lambda}\mu\tilde{\mu}$. En absence de $\tilde{\mu}$, il convient de fusionner les règles $(\rightarrow)$ et $(\tilde{\mu})$ du calcul $\overline{\lambda}\mu\tilde{\mu}$ en une nouvelle règle $(\rightarrow^\beta)$ (ce qui correspond par ailleurs à l'action de la β-réduction du calcul $\lambda\mu$ qui dans les faits inclut la substitution). Cette restriction correspond au calcul $\overline{\lambda}\mu_n$ défini en section 3.2.5.

Ainsi, si on considère $^{\mathsf{N}}$ allant du fragment η_μ^{app}-expansé du calcul $\lambda\mu$ vers le fragment sans $(\tilde{\mu})$ du calcul $\overline{\lambda}\mu\tilde{\mu}$, on obtient un isomorphisme. En particulier, la clause $(v\,v')^{\mathsf{N}}$ de la définition de $^{\mathsf{N}}$ devient inutile.

Proposition 32 *La traduction $^{\mathsf{N}}$ constitue un isomorphisme entre le calcul $\lambda\mu$ η_μ^{app}-expansé en appel par nom et le calcul $\overline{\lambda}\mu_n$ à réduction μ-atomique.*

PREUVE: On vérifie d'abord que $v[x \leftarrow v']^{\mathsf{N}} = v^{\mathsf{N}}[x \leftarrow v'^{\mathsf{N}}]$ (et notamment que $v[x \leftarrow v']^{\mathsf{N}}_{e[x\leftarrow v'^{\mathsf{N}}]} = v^{\mathsf{N}}_e[x \leftarrow v'^{\mathsf{N}}]$). Soit alors $(\lambda x.v)\,v'$ un rédex (β). Par la condition d'expansion η_μ^{app}, ce rédex est sous-terme d'un terme de la forme $\mu\alpha.[\beta]((\lambda x.v)\,v'\,v_1 \ldots v_n)$ dont la traduction est $\mu\alpha.\langle\lambda x.v^{\mathsf{N}} \| v'^{\mathsf{N}} \cdot v_1^{\mathsf{N}} \cdot \ldots \cdot v_n^{\mathsf{N}} \cdot \beta\rangle$. On vérifie que la traduction du réduit du rédex (β) est $\mu\alpha.\langle v^{\mathsf{N}}[x \leftarrow v'^{\mathsf{N}}] \| v_1^{\mathsf{N}} \cdot \ldots \cdot v_n^{\mathsf{N}} \cdot \beta\rangle$, lui-même obtenu par réduction $(\rightarrow^\beta)$.

Comme $[\beta]v^{\mathsf{N}} = v^{\mathsf{N}}_\beta$ et $[\beta](v\,v')^{\mathsf{N}} = v^{\mathsf{N}}_{v'^{\mathsf{N}}\cdot\beta}$ on en déduit que $v[\alpha \leftarrow [\beta](\Box\,v')]^{\mathsf{N}} = v^{\mathsf{N}}[\alpha \leftarrow v'^{\mathsf{N}} \cdot \beta]$ (et notamment que $v[\alpha \leftarrow [\beta](\Box\,v')]^{\mathsf{N}}_{e[\alpha\leftarrow v'^{\mathsf{N}}\cdot\beta]} = v^{\mathsf{N}}_e[\alpha \leftarrow v'^{\mathsf{N}} \cdot \beta]$) ce qui permet de vérifier la correspondance entre (μ_R) et (μ_{app}) dans tout contexte η_μ^{app}-expansé.

La correspondance entre les règles μ_{var} de chaque calcul est sans difficulté. ■

Nous donnons à titre indicatif le plongement inverse :

$$\begin{array}{lcl@{\qquad\qquad}rcl}
x^{\mathsf{N}^{-1}} & \triangleq & x & \alpha_v^{\mathsf{N}^{-1}} & \triangleq & [\alpha]v \\
(\lambda x.v)^{\mathsf{N}^{-1}} & \triangleq & \lambda x.v^{\mathsf{N}^{-1}} & (v' \cdot E)_v^{\mathsf{N}^{-1}} & \triangleq & E^{\mathsf{N}^{-1}}_{v\,(v'^{\mathsf{N}^{-1}})} \\
(\mu\alpha.c)^{\mathsf{N}^{-1}} & \triangleq & \mu\alpha.c^{\mathsf{N}^{-1}} & & &
\end{array}$$

$$\langle v \| E\rangle^{\mathsf{N}^{-1}} \triangleq E^{\mathsf{N}^{-1}}_{v^{\mathsf{N}^{-1}}}$$

3.13.4 Conditions d'un isomorphisme entre les calculs $\lambda\mu$ et $\overline{\lambda}\mu\tilde{\mu}$ en appel par valeur

Nous essayons dans cette section d'établir un lien entre le calcul $\lambda\mu$ en appel par valeur (gauche-droite) et le calcul $\overline{\lambda}\mu\tilde{\mu}$. Le premier problème est de caractériser la théorie du calcul $\lambda\mu$ en appel par valeur (gauche-droite). En particulier, l'ensemble de règles (β_v), (μ_L), (μ_{Rv}) et (μ_{var}) donné en section 3.13.1 est suffisant pour évaluer des termes clos en des expressions de la forme $\mu\alpha.[\alpha]V$ ou V, mais pas suffisant pour évaluer sous des abstractions (ou plus généralement des termes ouverts). Par exemple, l'expression $(\lambda x.\lambda y.x)\,(z_1\,z_2)\,z_3$ est normale pour (β_v), (μ_L), (μ_{Rv}) et (μ_{var}) mais l'on pourrait encore transporter z_3 vers le λy qui l'attend et obtenir $(\lambda x.x)\,(z_1\,z_2)$ qui lui-même pourrait encore être simplifié en $z_1\,z_2$. C'est du moins ce que nous apprennent les axiomatiques complètes du λ-calcul, ces axiomatiques qui sont complètes vis à vis de la traduction CPS de l'appel par valeur. On va voir que cette notion de complétude coïncide aussi avec ce que nous apprend le calcul $\overline{\lambda}\mu\tilde{\mu}$ par valeur.

Une axiomatique du calcul $\lambda\mu$ par valeur (gauche-droite)

Il n'existe pas à notre connaissance de description précise d'une axiomatisation complète du $\lambda\mu$ non typé. En adaptant les résultats de Selinger [Sel03] sur l'axiomatisation de la théorie en appel par valeur (gauche-droite) d'un sur-ensemble de la variante typée de de Groote [dG94] et Ong [Ong96] du calcul $\lambda\mu$ de Parigot, ainsi que les résultats de Sabry et Felleisen [SF93] (calcul non typé) et Hofmann [Hof95] (calcul typé) sur l'axiomatisation de la théorie en appel par valeur du λ-calcul étendu avec *callcc* et *abort*, et enfin de l'axiomatique de Moggi [Mog88] pour le λ-calcul, nous arrivons à l'axiomatique suivante que nous admettrons complète vis à vis de la traduction CPS du λ-calcul en appel par valeur gauche-droite vers le λ-calcul en appel par nom équipé de β et η.

La définition de cette axiomatique repose sur les notions de **contexte d'évaluation par valeur**, noté E_v, et de **brique de contexte d'évaluation par valeur**, notée F_v, dans le calcul $\lambda\mu$

$$\begin{array}{lcl} F_v & ::= & \Box\, v \mid (\lambda x.v)\, \Box \\ G_v & ::= & \Box \mid G_v\, v \\ E_v & ::= & G_v \mid (\lambda x.v)\, G_v \end{array}$$

Les notations $F_v[v]$, $E_v[v]$ désignent alors les termes reconstruits par substitution de v dans les contextes F_v et E_v. On donne les règles de réduction

$$\begin{array}{llcl} (\beta_v) & (\lambda x.v)\, V & \to & v[x \leftarrow V] \\ (\mu_v) & F_v[\mu\alpha.c] & \to & \mu\alpha.c[\alpha \leftarrow [\alpha]F_v] \\ (\mu_{var}) & [\alpha]\mu\beta.v & \to & v[\beta \leftarrow [\alpha]\Box] \\ (let_{lift}) & F_v[(\lambda x.v)\, v'] & \to & (\lambda x.F_v[v])\, v' \\ (\mu_{let}) & (\lambda x.\mu\alpha.[\beta]v)\, v' & \to & \mu\alpha.[\beta]((\lambda x.v)\, v') \\ (\mu_v^\eta) & v\, \mu\alpha.c & \to & (\lambda x.\mu\alpha.c[\alpha \leftarrow [\alpha](x\, \Box)])\, v \end{array}$$

et les règles d'η-conversion

$$\begin{array}{llcll} (\eta_v) & \lambda x.(V\, x) & \to & V & x \text{ non libre dans } V \\ (\eta_\mu) & \mu\alpha.[\alpha]v & \to & v & \alpha \text{ non libre dans } V \\ (\eta_{let}) & (\lambda x.E_v[x])\, v & \to & E_v[v] & x \text{ non libre dans } E_v \end{array}$$

Nous ne connaissons pas de présentation d'axiomatiques complètes du calcul $\lambda\mu$ par valeur dont les équations soient orientées en un ensemble confluent. Nous conjecturons que le système orienté ci-dessus est confluent et, que dans le cas typé, il est fortement normalisant. En particulier, la règle (μ_v^η) est redondante et ne sert qu'à la confluence (elle est dérivable des règles (μ_v), (η_v) et (η_{let})).

Signalons quelques connexions avec les axiomatiques de Sabry et Felleisen, de Selinger, et de Hofmann :

- La règle (η_{let}), avec l'aide de (η_v) permet de dériver les axiomes (β_{flat}), (β_Ω) et (β_{id}) de Sabry et Felleisen, les axiomes (id) et (let_{app}) de Moggi (repris par Selinger) et les axiomes Ass et App de Hofmann.
- La règle (η_μ) correspond à l'axiome éponyme de Selinger, à l'axiome $(\mathcal{C}_{elim})$ de Sabry et Felleisen et à l'axiome $\mathcal{C}$-App de Hofmann.
- La règle (let_{lift}) est équivalente à l'axiome (β_{lift}) de Sabry et Felleisen et elle permet de dériver les axiomes $(comp)$ de Moggi (repris par Selinger).
- La règle (μ_{var}) permet de dériver l'axiome (β_μ) de Selinger.
- La règle (μ_v) permet de dériver l'axiome (ζ) de Selinger et l'axiome $\mathcal{C}$-Nat de Hofmann.

Le rôle du `let-in`

Il est d'usage d'interpréter une expression $(\lambda x.v)\, v'$ du λ-calcul en appel par valeur comme un **let** $x = v'$ **in** v. Dans un souci de décrire les formes normales d'une manière telle que la propriété de la sous-formule soit vérifiée dans un cadre typé, nous ajoutons un opérateur `let-in` primitif au calcul. À la différence d'une forme normale de la forme $(\lambda x.v)\, v'$ qui introduit un objet de type fonctionnel (de type le type de x flèche le type de v) alors même que le résultat ne mentionne pas ce type (perte de la propriété de la sous-formule), la forme normale **let** $x = v'$ **in** v ne nécessite pas d'introduire des types plus complexes[6] que ceux figurant dans le jugement (en particulier le type de x sera un sous-type d'une des variables de v bloquant son évaluation). L'approche poussant à considérer la construction `let-in` comme primitive, qui est en accord avec ce que nous enseigne le calcul $\overline{\lambda}\mu\tilde{\mu}$, nous mène au calcul $\lambda\mu_{\texttt{let}}$ suivant :

[6] Moggi avance un argument comparable pour justifier le `let-in` primitif : celui-ci existe de manière autonome (dans les modèles catégoriques du calcul) sans que l'abstraction et l'application soient forcément présents.

$$
\begin{array}{lcl}
V & ::= & x \mid \lambda x.v \\
v & ::= & V \mid v\ v \mid \mu\alpha.c \mid \mathbf{let}\ x = v\ \mathbf{in}\ v \\
c & ::= & [\alpha]v \\
\\
F_v & ::= & \Box v \mid \mathbf{let}\ x = \Box\ \mathbf{in}\ v \\
G_v & ::= & \Box \mid G_v\ v \\
E_v & ::= & G_v \mid \mathbf{let}\ x = G_v\ \mathbf{in}\ v
\end{array}
$$

$$
\begin{array}{llcll}
(\beta_{let}) & (\lambda x.v')\ v & \to & \mathbf{let}\ x = v\ \mathbf{in}\ v' & \\
(let_v) & \mathbf{let}\ x = V\ \mathbf{in}\ v & \to & v[x \leftarrow V] & \\
(\mu_v) & F_v[\mu\alpha.v] & \to & \mu\alpha.v[\alpha \leftarrow [\alpha]F_v] & \\
(\mu_{var}) & [\alpha]\mu\beta.v & \to & v[\beta \leftarrow [\alpha]\Box] & \\
(let_{lift}) & F_v[\mathbf{let}\ x = v'\ \mathbf{in}\ v] & \to & \mathbf{let}\ x = v'\ \mathbf{in}\ F_v[v] & \\
(\mu_{let}) & (\mathbf{let}\ x = v'\ \mathbf{in}\ \mu\alpha.[\beta]v) & \to & \mu\alpha.[\beta]\mathbf{let}\ x = v'\ \mathbf{in}\ v & \\
(\mu_v^\eta) & v\ \mu\alpha.c & \to & \mathbf{let}\ x = v\ \mathbf{in}\ \mu\alpha.c[\alpha \leftarrow [\alpha](x\,\Box)] & \\
\\
(\eta_v) & \lambda x.(V\ x) & \to & V & x \text{ non libre dans } V \\
(\eta_\mu) & \mu\alpha.[\alpha]v & \to & v & \alpha \text{ non libre dans } V \\
(\eta_{let}) & \mathbf{let}\ x = v\ \mathbf{in}\ E_v[x] & \to & E_v[v] & x \text{ non libre dans } E_v
\end{array}
$$

où la règle (β_v) a profité de la nouvelle règle (let_v) pour être décomposée en une nouvelle règle (β_{let}) et (let_v).

Vers une connexion avec le calcul $\overline{\lambda}\mu\tilde{\mu}$ par valeur

Pour simplifier la connexion entre les calculs $\lambda\mu$ et $\overline{\lambda}\mu\tilde{\mu}$ par valeur (et en particulier sa restriction sur les contextes applicatifs spécifique à l'appel par valeur), nous restreignons la syntaxe du $\lambda\mu$-calcul en forçant les applications à être de la forme $v\,V$. Cette restriction est obtenue en appliquant systématiquement la règle (μ_v^η) partout où elle est applicable. Le calcul obtenu reste stable par réduction si bien que (μ_v^η) peut disparaître du système de réduction. On nomme ce calcul $\lambda\mu^\eta_{\mathtt{let}}$.

$$
\begin{array}{lcl}
V & ::= & x \mid \lambda x.v \\
v & ::= & V \mid v\,V \mid \mu\alpha.c \mid \mathbf{let}\ x = v\ \mathbf{in}\ v \\
c & ::= & [\alpha]v \\
\\
F_v & ::= & \mathbf{let}\ x = \Box\ \mathbf{in}\ v \\
G_v & ::= & \Box \mid G_v\,V \\
E_v & ::= & G_v \mid \mathbf{let}\ x = G_v\ \mathbf{in}\ v
\end{array}
$$

$$
\begin{array}{llcll}
(\beta_{let}) & (\lambda x.v')\ v & \to & \mathbf{let}\ x = v\ \mathbf{in}\ v' & \\
(let) & \mathbf{let}\ x = V\ \mathbf{in}\ v & \to & v[x \leftarrow V] & \\
(\mu_v) & F_v[\mu\alpha.v] & \to & \mu\alpha.v[\alpha \leftarrow [\alpha]F_v] & \\
(\mu_{var}) & [\alpha]\mu\beta.v & \to & v[\beta \leftarrow [\alpha]\Box] & \\
(let_{lift}) & F_v[\mathbf{let}\ x = l\ \mathbf{in}\ v] & \to & \mathbf{let}\ x = l\ \mathbf{in}\ F_v[v] & \\
(\mu_{let}) & (\mathbf{let}\ x = v'\ \mathbf{in}\ \mu\alpha.[\beta]v) & \to & \mu\alpha.[\beta]\mathbf{let}\ x = v'\ \mathbf{in}\ v & \\
\\
(\eta_v) & \lambda x.(V\ x) & \to & V & x \text{ non libre dans } V \\
(\eta_\mu) & \mu\alpha.[\alpha]v & \to & v & \alpha \text{ non libre dans } V \\
(\eta_{let}) & \mathbf{let}\ x = v\ \mathbf{in}\ E_v[x] & \to & E_v[v] & x \text{ non libre dans } E_v
\end{array}
$$

Saturation en rédex η_μ

Pour permettre un isomorphisme avec le $\overline{\lambda}\mu\tilde{\mu}$ par valeur, il convient de contraindre encore un peu plus le $\lambda\mu$-calcul en expansant au moyen de la règle η_μ les expressions de la forme **let** $x = v$ **in** v' ou $v\,V_1 \ldots V_n$ qui sont sous-termes immédiats d'une abstraction. Cette expansion, que nous nommons $\eta_\mu^{let-app}$-expansion, est comparable à la η_μ^{app}-expansion qui avait été considérée pour permettre un isomorphisme dans le cas appel par nom.

Autrement dit, un terme du calcul $\lambda\mu$ avec `let-in` est $\eta_\mu^{let-app}$-expansé si on ne peut remplacer aucun sous-terme d'un terme v de la forme **let** $x = v'$ **in** v'' ou $v\,V_1 \ldots V_n$ par $\mu\alpha.[\alpha]v$ (pour α frais) sans créer de rédex (autre que η_μ). Dans la pratique, cela veut dire qu'un terme est $\eta_\mu^{let-app}$-expansé s'il ne contient aucun sous-terme de la forme $\lambda x.$**let** $y = v$ **in** v' ou de la forme $\lambda x.(v\,V_1 \ldots V_n)$. Le calcul résultant, le $\lambda\mu^\eta_{\mathtt{let}}\ \eta_\mu^{let-app}$-expansé, calcul où η_μ n'apparaît plus, est décrit par la syntaxe suivante :

$$\begin{array}{lcl}
V & ::= & x \mid \lambda x.t \\
t & ::= & V \mid \mu\alpha.c \\
v & ::= & t \mid v\,V \mid \textbf{let } x = v \textbf{ in } v \\
c & ::= & [\alpha]v \\
\\
F_v & ::= & \square\,V \mid \textbf{let } x = \square \textbf{ in } v \\
G_v & ::= & \square \mid G_v\,V \\
E_v & ::= & G_v \mid \textbf{let } x = G_v \textbf{ in } v
\end{array}$$

$$\begin{array}{llcll}
(\beta_{let}) & (\lambda x.t)\,v & \rightarrow & \textbf{let } x = v \textbf{ in } t & \\
(let_v) & \textbf{let } x = V \textbf{ in } v & \rightarrow & v[x \leftarrow V] & \\
(\mu_v) & F_v[\mu\alpha.v] & \rightarrow & \mu\alpha.v[\alpha \leftarrow [\alpha]F_v] & \\
(\mu_{var}) & [\alpha]\mu\beta.v & \rightarrow & v[\beta \leftarrow [\alpha]\square] & \\
(let_{lift}) & F_v[\textbf{let } x = l \textbf{ in } v] & \rightarrow & \textbf{let } x = l \textbf{ in } F_v[v] & \\
(\mu_{let}) & (\textbf{let } x = v' \textbf{ in } \mu\alpha.[\beta]v) & \rightarrow & \mu\alpha.[\beta]\textbf{let } x = v' \textbf{ in } v & \\
\\
(\eta_v^\mu) & \lambda x.\mu\alpha.[\alpha](V\,x) & \rightarrow & V & x \text{ et } \alpha \text{ non libres dans } V \\
(\eta_{let}) & \textbf{let } x = v \textbf{ in } E_v[x] & \rightarrow & E_v[v] & x \text{ non libre dans } E_v
\end{array}$$

On peut maintenant décrire le plongement $^\vee$.

$$\begin{array}{lcl}
x^\vee & \triangleq & x \\
(\lambda x.t)^\vee & \triangleq & \lambda x.t^\vee \\
(\mu\alpha.c)^\vee & \triangleq & \mu\alpha.c^\vee \\
\\
([\alpha]v)^\vee & \triangleq & v^\vee_\alpha \\
\\
t^\vee_e & \triangleq & \langle t^\vee \| e\rangle \\
(v\,V)^\vee_e & \triangleq & v^\vee_{V^\vee \cdot e} \\
(\textbf{let } x = v \textbf{ in } v')^\vee_e & \triangleq & \langle \mu\alpha.\overline{\langle v^\vee \| \tilde{\mu}x.v'^\vee_\alpha\rangle} \| e\rangle \quad \text{pour } \alpha \text{ frais et si } e \text{ différent d'une variable} \\
(\textbf{let } x = v \textbf{ in } v')^\vee_\alpha & \triangleq & v^\vee_{\tilde{\mu}x.v'^\vee_\alpha}
\end{array}$$

dont on vérifie que l'image est bien dans le calcul $\overline{\lambda}\mu\tilde{\mu}_Q$.

Une analyse des formes normales du calcul $\lambda\mu^\eta_{\mathtt{let}}\ \eta_\mu^{let-app}$-expansé montre qu'elles sont décrites par la syntaxe suivante :

$$\begin{array}{lcl}
V & ::= & x \mid \lambda x.t \\
t & ::= & V \mid \mu\alpha.c \\
v & ::= & V \mid \textbf{let } x = l \textbf{ in } v \mid l \\
l & ::= & x\,V \mid l\,V \\
c & ::= & [\alpha]v
\end{array}$$

On observe que les formes normales sont respectées par le plongement $^{\mathsf{V}}$, y compris pour les réductions η.

Proposition 33 *v est normal dans le calcul $\lambda\mu^{\eta}_{\mathtt{let}}$ $\eta^{let-app}_{\mu}$-expansé ssi v^{V} est normal dans le calcul $\overline{\lambda}\mu\tilde{\mu}_v$.*

En particulier, toute expression normale de la forme

$$\lambda x_1 \dots x_n.\mu\alpha.[\beta]\textbf{let } y_1 = z_1\, V^1\, V^1_1 \dots V^1_{n_1} \textbf{ in } \dots \textbf{ let } y_m = z_m\, V^m\, V^m_1 \dots V^m_{n_m} \textbf{ in } z\, V'_1 \dots V'_p$$

est traduite en

$$\lambda x_1 \dots x_n.\mu\alpha.\langle z_1 \| V^1 \cdot V^1_1 \cdot \dots \cdot V^1_{n_1} \cdot \dots \cdot \tilde{\mu} y_1.\overline{\langle z_m \| V^m \cdot V^m_1 \cdot \dots \cdot V^m_{n_m} \cdot \dots \cdot \tilde{\mu} y_m.\overline{\langle z \| V'_1 \cdot \dots \cdot V'_p \cdot \beta\rangle}\rangle}\rangle\,.$$

Commutation du `let-in` et retardement du μ

Le plongement $^{\mathsf{V}}$ n'est pas injectif (et donc pas bijectif sur son image dans le calcul $\mu\tilde{\mu}_v^{\to^\eta}$) mais il le deviendrait si on retirait la clause **let** $x = v$ **in** ${v'}^{\mathsf{V}}_e$ pour e différent d'une variable.

L'objectif de cette clause est de traiter les rédex de la forme **let** $x =$ (**let** $y = v$ **in** v') **in** v'' ou (**let** $y = v$ **in** v') V et de les différencier de leur contracta **let** $y = v$ **in** **let** $x = v'$ **in** v'' ou **let** $y = v$ **in** $(v'\,V)$. Nous n'avons pas réussi à définir une traduction qui soit à la fois bijective sur les formes normales et respectueuse de la simulation des rédex $F_c[$**let** $x = v$ **in** $v']$.

Pour rétablir la bijectivité de la traduction, une possibilité est d'étendre le calcul $\mu\tilde{\mu}_v^{\to^\eta}$ avec une copie de l'opérateur μ spécifiquement dédiée à la simulation des rédex de la forme $F_c[$**let** $x = v$ **in** $v']$. Cette copie de μ, notée $\bar{\mu}$, est appelée **opérateur de retardement du** μ. Cet opérateur ne lie des variables que linéairement, si bien que l'on peut se contenter pour représenter cette liaison d'un unique nom de variable que l'on note $\star$. La syntaxe élargie des commandes est

$$c \quad ::= \quad \langle v \| e\rangle \mid \langle \bar{\mu}\star.c \| e\rangle$$

En plus de la linéarité de $\star$, l'expression $\langle \bar{\mu}\star.c \| e\rangle$ obéit à des contraintes sur la forme de c et e. Soit e un contexte d'évaluation. On définit la conclusion de e, notée $\mathsf{ccl}(e)$ par les clauses suivantes

$$\begin{array}{lcll}
\mathsf{ccl}(\alpha) & \triangleq & \alpha & \text{(y compris si } \alpha \text{ est } \star) \\
\mathsf{ccl}(V \cdot e) & \triangleq & \mathsf{ccl}(e) & \\
\mathsf{ccl}(\tilde{\mu}x.\langle v \| e\rangle) & \triangleq & \mathsf{ccl}(e) & \\
\mathsf{ccl}(\tilde{\mu}x.\langle \bar{\mu}\star.c \| e\rangle) & \triangleq & \mathsf{ccl}(e) &
\end{array}$$

Si de plus, e est de la forme $V_1 \cdot \dots \cdot V_n \cdot \alpha$, on dit que e est applicatif.

On peut maintenant caractériser l'image de $^{\mathsf{V}}$ comme l'extension du calcul $\mu\tilde{\mu}_v^{\to^\eta}$ avec les termes de la forme $\langle \bar{\mu}\star.\overline{\langle v \| e'\rangle} \| e\rangle$ sous la condition que e n'est pas une variable et que e' est de conclusion $\star$ sans être applicatif. On qualifie le calcul obtenu de calcul $\mu\tilde{\mu}_v^{\to^\eta}$ à opérateur de retardement du μ.

On peut alors modifier le plongement $^{\mathsf{V}}$ en remplaçant la clause définissant (**let** $x = v$ **in** $v')^{\mathsf{V}}_e$ pour e différent d'une variable par

$$(\textbf{let } x = v \textbf{ in } v')^{\mathsf{V}}_e \quad \triangleq \quad \langle \bar{\mu}\star.\overline{\langle v^{\mathsf{V}} \| \tilde{\mu}x.{v'}^{\mathsf{V}}_{\star}\rangle} \| e\rangle \quad \text{si } e \text{ différent d'une variable}$$

On peut alors maintenant définir la traduction réciproque.

$$\begin{array}{lcllcl}
x^{\mathsf{V}^{-1}} & \triangleq & x & \alpha_v^{\mathsf{V}^{-1}} & \triangleq & v \\
(\lambda x.v)^{\mathsf{V}^{-1}} & \triangleq & \lambda x.v^{\mathsf{V}^{-1}} & (V \cdot e)_v^{\mathsf{V}^{-1}} & \triangleq & e^{\mathsf{V}^{-1}}_{v\,(V^{\mathsf{V}^{-1}})} \\
(\mu\alpha.c)^{\mathsf{V}^{-1}} & \triangleq & \mu\alpha.c^{\mathsf{V}^{-1}} & (\tilde{\mu}x.\langle v' \| e\rangle)_v^{\mathsf{V}^{-1}} & \triangleq & \textbf{let } x = v \textbf{ in } e^{\mathsf{V}^{-1}}_{v'}
\end{array}$$

$$\begin{array}{lcll}
\langle v \| e\rangle^{\mathsf{V}^{-1}} & \triangleq & [\alpha]e^{\mathsf{V}^{-1}}_{v^{\mathsf{V}^{-1}}} & \text{où } \alpha = \mathsf{ccl}(e) \\
\langle \bar{\mu}\star.\overline{\langle v \| e'\rangle} \| e\rangle^{\mathsf{V}^{-1}} & \triangleq & e^{\mathsf{V}^{-1}}_{{e'}^{\mathsf{V}^{-1}}_{v^{\mathsf{V}^{-1}}}} &
\end{array}$$

où par construction de $\bar{\mu}$, $\mathsf{ccl}(e')$ est nécessairement $\star$ dans la dernière clause.

Comparaison des mécanismes de commutation de contextes

Maintenant que nous avons caractérisé l'image de la traduction, il reste à faire correspondre les règles de réduction. La principale difficulté est dans la règle de commutation de contexte autour d'un μ ou d'un `let-in`.

En effet, les appels à α sont placés de manière très différente entre le calcul $\lambda\mu$ et son image dans le calcul $\overline{\lambda}\mu\tilde{\mu}$: dans le calcul $\lambda\mu$, les $[\alpha]$ sont mitoyens des $\mu\beta$ alors que dans le calcul $\overline{\lambda}\mu\tilde{\mu}$, les α sont à l'opposé, au plus profond des contextes d'évaluation. Ainsi, toute substitution de α par un morceau de contexte F_c dans le calcul $\lambda\mu$ va mettre en jeu plusieurs étapes de commutation de `let-in` avant que ce contexte n'atteigne la position qu'il aurait atteint instantanément si la substitution avait eu lieu dans le calcul $\overline{\lambda}\mu\tilde{\mu}$. Par exemple,

$$\begin{array}{ll} & \mu\alpha.\langle z_1\|V^1\cdot\ldots\cdot\tilde{\mu}y_1.\langle\overline{z_m\|V^m\cdot\ldots\cdot\tilde{\mu}y_m.\langle\overline{z\|V'_1\cdot\ldots\cdot V'_p\cdot\beta}\rangle}\rangle\rangle[\beta\leftarrow V'\cdot\beta] \\ \to_\mu & \mu\alpha.\langle z_1\|V^1\cdot\ldots\cdot\tilde{\mu}y_1.\langle\overline{z_m\|V^m\cdot\ldots\cdot\tilde{\mu}y_m.\langle\overline{z\|V'_1\cdot\ldots\cdot V'_p\cdot V'\cdot\beta}\rangle}\rangle\rangle \end{array}$$

tandis que

$$\begin{array}{ll} & (\mu\alpha.[\beta]\mathbf{let}\ y_1 = z_1\,V^1_1\ldots V^1_{n_1}\ \mathbf{in}\ \ldots\ \mathbf{let}\ y_m = z_m\,V^m_1\ldots V^m_{n_m}\ \mathbf{in}\ z\,V'_1\ldots V'_p)[\beta\leftarrow[\beta](\Box\,V')] \\ \to_{\mu_v} & \mu\alpha.[\beta](\mathbf{let}\ y_1 = z_1\,V^1_1\ldots V^1_{n_1}\ \mathbf{in}\ \ldots\ \mathbf{let}\ y_m = z_m\,V^m_1\ldots V^m_{n_m}\ \mathbf{in}\ z\,V'_1\ldots V'_p)\,V' \\ \xrightarrow{*}_{\mu_{let}} & \mu\alpha.[\beta]\mathbf{let}\ y_1 = z_1\,V^1_1\ldots V^1_{n_1}\ \mathbf{in}\ \ldots\ (\mathbf{let}\ y_m = z_m\,V^m_1\ldots V^m_{n_m}\ \mathbf{in}\ z\,V'_1\ldots V'_p)\,V' \\ \to_{\mu_{let}} & \mu\alpha.[\beta]\mathbf{let}\ y_1 = z_1\,V^1_1\ldots V^1_{n_1}\ \mathbf{in}\ \ldots\ \mathbf{let}\ y_m = z_m\,V^m_1\ldots V^m_{n_m}\ \mathbf{in}\ z\,V'_1\ldots V'_p\,V' \end{array}$$

Pour simuler la substitution du calcul $\lambda\mu$, il faut donc décomposer l'action de la règle (μ) du calcul $\overline{\lambda}\mu\tilde{\mu}$ en une multitude d'étapes, et ce sans confusion entre l'étape (μ_v) et les étapes (μ_{let}). C'est là que $\bar{\mu}$ va intervenir : μ symbolisera les étapes (μ_v) tandis que les étapes (μ_{let}) seront explicitées par un $\bar{\mu}$ qui de fait sera à liaison linéaire.

Commençons par simuler (μ). Nous avons besoin pour cela de caractériser les sous-calculs qui envoient ultimement leur résultat vers α pour α une variable de contexte d'évaluation donnée. Si e est un contexte d'évaluation de conclusion α, on définit la substitution dans e de la conclusion α de e par α', notée $e[\alpha\rightsquigarrow\alpha']$, comme l'unique e' de conclusion α' tel que $e_x^{\mathsf{V}^{-1}} = {e'}_x^{\mathsf{V}^{-1}}$ pour un x arbitraire.

La substitution **retardée** de α par e dans c, notée $c[\alpha\hookleftarrow e]$, diffère de la substitution standard par l'expansion que crée la traversée d'un contexte $\tilde{\mu}x.c$ de conclusion la variable substituée. La définition est la suivante :

$$\begin{array}{lcll} \langle\bar{\mu}\star.c\|e'\rangle[\alpha\hookleftarrow e] & \triangleq & \langle\bar{\mu}\star.c[\alpha\hookleftarrow e]\|e'[\alpha\hookleftarrow e]\rangle & \\ \langle v\|e'\rangle[\alpha\hookleftarrow e] & \triangleq & \langle\bar{\mu}\star.\langle\overline{v[\alpha\hookleftarrow e]\|e'[\alpha\rightsquigarrow\star][\alpha\hookleftarrow e]}\rangle\|e\rangle & \text{si } e' \text{ non applicatif de conclusion } \alpha \\ \langle v\|e'\rangle[\alpha\hookleftarrow e] & \triangleq & \langle v[\alpha\hookleftarrow e]\|e'[\alpha\hookleftarrow e]\rangle & \text{sinon} \\ & & & \\ (\mu\beta.c)[\alpha\hookleftarrow e] & \triangleq & \mu\beta.c[\alpha\hookleftarrow e] & \\ (\lambda x.v)[\alpha\hookleftarrow e] & \triangleq & \lambda x.v[\alpha\hookleftarrow e] & \\ x[\alpha\hookleftarrow e] & \triangleq & x & \\ & & & \\ (\tilde{\mu}x.c)[\alpha\hookleftarrow e] & \triangleq & \tilde{\mu}x.c[\alpha\hookleftarrow e] & \\ (v\cdot e')[\alpha\hookleftarrow e] & \triangleq & v[\alpha\hookleftarrow e]\cdot e'[\alpha\hookleftarrow e] & \\ \beta[\alpha\hookleftarrow e] & \triangleq & \beta & \\ \alpha[\alpha\hookleftarrow e] & \triangleq & e & \end{array}$$

où les variables liantes sont choisies afin d'éviter toute capture.

On peut alors définir les règles suivantes en remplacement de la règle (μ).

$$\begin{array}{llcll} (\mu_{app}) & \langle\mu\alpha.c\|V\cdot e\rangle & \xrightarrow{h} & \langle\mu\alpha.c[\alpha\hookleftarrow V\cdot\alpha]\|e\rangle & \\ (\mu_{\tilde{\mu}}) & \langle\mu\alpha.c\|\tilde{\mu}x.c'\rangle & \xrightarrow{h} & \langle\mu\alpha.c[\alpha\hookleftarrow(\tilde{\mu}x.c')[\beta\rightsquigarrow\alpha]]\|\beta\rangle & \text{si } \tilde{\mu}x.c' \text{ de conclusion } \beta \\ (\mu_{var}) & \langle\mu\alpha.c\|\beta\rangle & \to & c[\alpha\leftarrow\beta] & \text{si } \beta \text{ différent de } \star \end{array}$$

Pour l'opérateur $(\bar{\mu})$, les règles sont

$$\begin{array}{llcll}
(\bar{\mu}_{app}) & \langle \bar{\mu}\star.\langle\overline{v\|e}\rangle\|V\cdot e'\rangle & \xrightarrow{h} & \langle \bar{\mu}\star.\langle\overline{v\|e[\star\hookleftarrow V\cdot\star]}\rangle\|e'\rangle & \text{si } e' \text{ différent d'une variable}\\
(\bar{\mu}) & \langle \bar{\mu}\star.\langle\overline{v\|e}\rangle\|e'\rangle & \xrightarrow{h} & \langle v\|e[\star\hookleftarrow e']\rangle & \text{si } e' \text{ de la forme } V\cdot\beta \text{ ou } \tilde{\mu}x.c\\
(\eta^{\star}_{\tilde{\mu}}) & \langle \bar{\mu}\star.\langle\overline{v\|\tilde{\mu}x.\langle\overline{x\|e'}\rangle}\rangle\|e\rangle & \xrightarrow{h} & \langle v\|e'[\star\hookleftarrow e]\rangle & \text{si } x \text{ non libre dans } e'
\end{array}$$

En particulier, on vérifie que toute instance de (μ) peut se décomposer en une suite de réductions (μ_{app}), $(\mu_{\tilde{\mu}})$ et finalement (μ_{var}), et de réductions $(\bar{\mu}_{app})$ et $(\bar{\mu})$ pour les rédex créés par la substitution. On appelle l'ensemble des règles (μ_{app}), $(\mu_{\tilde{\mu}})$, (μ_{var}), $(\bar{\mu}_{app})$ et $(\bar{\mu})$ **règles de substitution incrémentale et retardée des contextes d'évaluation**. Quant à la règle $(\eta^{\star}_{\tilde{\mu}})$, elle correspond à une interaction particulière de la règle $(\eta_{\tilde{\mu}})$ avec l'opérateur $\bar{\mu}$.

Il reste le cas de la règle (μ_{let}) dont l'image dans le calcul $\mu\tilde{\mu}_v^{\rightarrow^{\eta}}$ donne les règles

$$\begin{array}{llcl}
(\mu_{up}) & \langle v'\|\tilde{\mu}x.\langle\overline{\mu\alpha.c\|\gamma}\rangle\rangle & \xrightarrow{h} & \langle \mu\alpha.\langle\overline{v'\|\tilde{\mu}x.c}\rangle\|\gamma\rangle\\
(\mu^{\star}_{up}) & \langle \bar{\mu}\star.\langle\overline{v'\|\tilde{\mu}x.\langle\overline{\mu\alpha.c\|\star}\rangle}\rangle\|e\rangle & \xrightarrow{h} & \langle \mu\alpha.\langle\overline{v'\|\tilde{\mu}x.c}\rangle\|e\rangle
\end{array}$$

pour laquelle α est choisi de manière à éviter toute capture dans v'. Il reste aussi le cas de la règle (let_v) qui peut détruire un $\bar{\mu}$:

$$(\tilde{\mu}^{\star}) \quad \langle \bar{\mu}\star.\langle\overline{V\|\tilde{\mu}x.\langle\overline{v\|e'}\rangle}\rangle\|e\rangle \quad \xrightarrow{h} \quad \langle v[x\leftarrow V]\|e'[\star\hookleftarrow e]\rangle$$

On a alors

Proposition 34 *Le plongement* $^{\vee}$ *est un isomorphisme entre le calcul* $\lambda\mu^{\eta}_{\mathtt{let}}$ $\eta_{\mu}^{let-app}$*-expansé et le calcul* $\mu\tilde{\mu}_v^{\rightarrow^{\eta}}$ *à opérateur de retardement du* μ *et équipé des règles de substitution incrémentale et retardée des contextes d'évaluation et des règles* $(\eta^{\star}_{\tilde{\mu}})$, (μ_{up}), $(\mu^{\star}_{up})$ *et* $(\tilde{\mu}^{\star})$.

En particulier, on a la correspondance suivante entre les règles :

(β_{let})	$(\rightarrow)$
(let_v)	$(\tilde{\mu})$ et $(\tilde{\mu}^{\star})$
(μ_v)	$(\mu_{\tilde{\mu}})$ et (μ_{app})
(μ_{var})	(μ_{var})
(let_{lift})	$(\bar{\mu})$ et $(\bar{\mu}_{app})$
(μ_{let})	(μ_{up}) et $(\mu^{\star}_{up})$
(η_v)	$(\eta^{R}_{\rightarrow v})$
(η_{let})	$(\eta_{\tilde{\mu}})$ et $(\eta^{\star}_{\tilde{\mu}})$

Commentaires

La complexité de l'analyse précédente montre à quel point la syntaxe stricte du calcul $\lambda\mu$ se prête mal à une définition simple de l'appel par valeur. Entre le calcul $\overline{\lambda}\tilde{\mu}_v$, qui admet une formulation relativement simple en syntaxe λ-calcul (cf la section 3.2.7) et le calcul $\lambda\mu$ de base, il y a un intermédiaire à trouver.

Notons que le cas intuitionniste est plus simple car seul $\bar{\mu}$ reste nécessaire pour la simulation (le μ du calcul $\overline{\lambda}\mu\tilde{\mu}_v$, qui servait à interprèter le μ du calcul $\lambda\mu$, perd son utilité). Il suffit alors de renommer $\bar{\mu}$ en μ pour que la simulation reste à l'intérieur de la syntaxe du calcul $\mu\tilde{\mu}_v^{\rightarrow^{\eta}}$ sans nécessiter d'opérateur supplémentaire[7].

[7] Dans le fragment intuitionniste, on a la correspondance suivante entre les règles :

(β_{let})	$(\rightarrow)$
(let_v)	$(\tilde{\mu})$ et $(\tilde{\mu}^{\star})$
(let_{lift})	$(\bar{\mu})$ et $(\bar{\mu}_{app})$
(η_v)	$(\eta^{R}_{\rightarrow v})$
(η_{let})	$(\eta_{\tilde{\mu}})$ et $(\eta^{\star}_{\tilde{\mu}})$

ce qui confirme que μ n'est plus utile et qu'un $\lambda^{\eta}_{\mathtt{let}}$ sans $\eta_{\mu}^{let-app}$-expansion est en correspondance avec le fragment intuitionniste de $\mu\tilde{\mu}_v^{\rightarrow^{\eta}}$.

3.14 Autres calculs symétriques et connexions

3.14.1 Le calcul symétrique de Filinski

Le λ-calcul symétrique de Filinski (SLC, pour « Symmetric Lambda-Calculus ») est (à notre connaissance) la première formulation d'un λ-calcul présentant une symétrie explicite entre « valeurs » et « continuations » et équipé de deux interprétations dénotationnelles symétriques l'une de l'autre, la première pour l'appel par nom et la seconde pour l'appel par valeur [Fil89a, Fil89b].

Le λ-calcul symétrique de Filinski lui-même n'est pas équipé d'un système de règles de réduction. Il est partiellement équipé d'égalités (deux versions duales de β et η) qui pourraient être orientées mais il n'y a pas de paire critique dont la déterminisation correspond au choix CBN ou CBV, comme c'est le cas dans le système $\mu\tilde{\mu}$. Malgré tout, l'analyse de la dualité y est remarquablement poussée, allant jusqu'à une dualité entre paire paresseuse du CBV et analyse de cas « stricte » du CBN, ou entre récursion et corécursion (cf sections 2.6 et 2.5.2 de [Fil89b]).

Le calcul contient trois catégories d'objets : les *valeurs*, les *continuations* et les *préfonctions* (« function templates » en anglais). Les premières correspondent respectivement aux termes et contextes d'évaluation du système $\mu\tilde{\mu}$ et sont symétriques l'un de l'autre. Les préfonctions sont invariantes par symétrie, comme les commandes du système $\mu\tilde{\mu}$, mais elles s'en distinguent par le fait que ce sont des « tuyaux », c'est-à-dire des termes à deux trous, destinés à être connectées d'un côté à une valeur et de l'autre à une continuation, alors que les commandes sont des expressions sans trou.

Le calcul a pour connecteurs le produit et son dual la somme, l'implication (notée $[A \to B]$ comme type de valeurs) et son dual la soustraction (notée $[A \leftarrow B]$ comme type de continuations), ainsi que le type unité et son dual le type vide.

Le calcul a deux classes de variables, l'une pour dénoter des valeurs, l'autre pour dénoter des continuations. Nous utiliserons dans la suite x, y, ... pour la première classe et α, β, ... pour la seconde classe (au lieu de x, x', ... et y, y', ... dans la présentation originelle).

Contrairement à la version originelle, nous ne prendrons pas en compte la gestion de valeurs primitives (telles que entiers, opérations arithmétiques, etc.).

Pour simplifier, nous ne considérons aussi les abstractions que sur des motifs non imbriqués. Par exemple le motif $((x, y), z)$ n'est pas pris en compte et la fonction $((x, y), z) \Rightarrow E$ devra s'écrire $(z', z) \Rightarrow ((x, y) \Rightarrow E \uparrow z')$. C'est une simplification du langage compatible avec les sémantiques tant CBN que CBV, et, de toutes façons, aucune règle (en dehors de la sémantique dénotationnelle) n'est donnée chez Filinski [Fil89b] pour évaluer les motifs avec imbrication.

La syntaxe ainsi modifiée du λ-calcul symétrique de Filinski est la suivante :

Préfonctions	F	$::=$	$X \Rightarrow E \mid \Xi \Leftarrow C \mid \overline{E} \mid \underline{C}$
Valeurs	E	$::=$	$x \mid F \uparrow E \mid () \mid (E, E) \mid \ulcorner F \urcorner$
Continuations	C	$::=$	$\alpha \mid C \downarrow F \mid \{\} \mid \{C, C\} \mid \llcorner F \lrcorner$
Motifs de valeur	X	$::=$	$x \mid () \mid (x, y)$
Motifs de continuation	Ξ	$::=$	$\alpha \mid \{\} \mid \{\alpha, \beta\}$

On peut associer aux termes un système de types simples dont les types sont définis par la grammaire suivante :

$$A, B ::= X \mid [A \Rightarrow B] \mid [B \Leftarrow A] \mid A \times B \mid A + B \mid unit \mid null$$

Le λ-calcul symétrique original de Filinski est considéré simplement typé même si la version non typée reste pertinente. Dans Filinski [Fil89a, Fil89b], les contextes de typage ne sont pas explicitement considérés, mais nous les explicitons ici. En particulier, les valeurs, continuations et préfonctions sont respectivement interprétées par des jugements de la forme $\Gamma \vdash E : A; \Delta$, et $\Gamma; C : A \vdash \Delta$, et $F : (\Gamma; A \vdash B; \Delta)$. Dans les jugements, les contextes sont considérés à l'ordre près. Les règles sont les suivantes :

$$\frac{x : A \text{ dans } \Gamma}{\Gamma \vdash x : A; \Delta} \qquad \frac{\alpha : A \text{ dans } \Delta}{\Gamma; \alpha : A \vdash \Delta}$$

$$\frac{F : (\Gamma; A \vdash B; \Delta) \quad \Gamma \vdash E : A; \Delta}{\Gamma \vdash F \uparrow E : B; \Delta} \qquad \frac{\Gamma; C : B \vdash \Delta \quad F : (\Gamma; A \vdash B; \Delta)}{\Gamma; C \downarrow F : A \vdash \Delta}$$

$$\frac{}{\Gamma \vdash () : \mathit{unit}; \Delta} \qquad \frac{}{\Gamma; \{\} : \mathit{null} \vdash \Delta}$$

$$\frac{\Gamma \vdash E_1 : A_1; \Delta \quad \Gamma \vdash E_2 : A_2; \Delta}{\Gamma \vdash (E_1, E_2) : A_1 \times A_2; \Delta} \qquad \frac{\Gamma; C_1 : A_1 \vdash \Delta \quad \Gamma; C_2 : A_2 \vdash \Delta}{\Gamma; \{C_1, C_2\} : A_1 + A_2 \vdash \Delta}$$

$$\frac{F : (\Gamma; A \vdash B; \Delta)}{\Gamma \vdash \ulcorner F \urcorner : [A \to B]; \Delta} \qquad \frac{F : (\Gamma; A \vdash B; \Delta)}{\Gamma; \llcorner F \lrcorner : [B \leftarrow A] \vdash \Delta}$$

$$\frac{\Gamma, x : A \vdash E : B; \Delta}{x \Rightarrow E : (\Gamma; A \vdash B; \Delta)} \qquad \frac{\Gamma; C : A \vdash \beta : B, \Delta}{\beta \Leftarrow C : (\Gamma; A \vdash B; \Delta)}$$

$$\frac{\Gamma \vdash E : B; \Delta}{() \Rightarrow E : (\Gamma; \mathit{unit} \vdash B; \Delta)} \qquad \frac{\Gamma; C : A \vdash \Delta}{\{\} \Leftarrow C : (\Gamma; A \vdash \mathit{null}; \Delta)}$$

$$\frac{\Gamma, x_1 : A_1, x_2 : A_2 \vdash E : B; \Delta}{(x_1, x_2) \Rightarrow E : (\Gamma; A_1 \times A_2 \vdash B; \Delta)} \qquad \frac{\Gamma; C : A \vdash \beta_1 : B_1, \beta_2 : B_2, \Delta}{\{\beta_1, \beta_2\} \Leftarrow C : (\Gamma; A \vdash B_1 + B_2; \Delta)}$$

$$\frac{\Gamma \vdash E : [A \to B]; \Delta}{\overline{E} : (\Gamma; A \vdash B; \Delta)} \qquad \frac{\Gamma; C : [B \leftarrow A] \vdash \Delta}{\underline{C} : (\Gamma; A \vdash B; \Delta)}$$

Notons que ce calcul (en fait d'inspiration catégorique) est hybride. La construction $F \uparrow E$ est typée par un Modus Ponens et la construction $C \downarrow F$ par son équivalent pour la soustraction. L'association de motifs aux déclarations du contexte de typage (tel que considérée dans la version originelle) n'existe pas non plus dans le système de typage du système $\mu\tilde{\mu}$ et de ses extensions.

On va maintenant plonger le λ-calcul symétrique de Filinski dans le système $\mu\tilde{\mu}$. On utilise deux noms de variables spéciaux, $\bullet$ pour les termes et $\blacksquare$ pour les contextes d'évaluation, pour désigner les « entrées » et « sorties » des préfonctions.

$$\begin{array}{lcl}
[\![x]\!] & \triangleq & x \\
[\![F \uparrow E]\!] & \triangleq & \mu\blacksquare.\langle [\![E]\!] \| \tilde{\mu} \bullet . [\![F]\!] \rangle \\
[\![()]\!] & \triangleq & \ltimes \\
[\![(E_1, E_2)]\!] & \triangleq & \mu\alpha.\langle [\![E_1]\!] \| \tilde{\mu} x_1.\overline{\langle [\![E_2]\!] \| \tilde{\mu} x_2.\overline{\langle (x_1, x_2) \| \alpha \rangle} \rangle} \rangle \\
[\![\ulcorner F \urcorner]\!] & \triangleq & \lambda(\bullet, \blacksquare).[\![F]\!] \\
\\
[\![\alpha]\!] & \triangleq & \alpha \\
[\![C \downarrow F]\!] & \triangleq & \tilde{\mu} \bullet .\langle \mu\blacksquare.[\![F]\!] \| [\![C]\!] \rangle \\
[\![\{\}]\!] & \triangleq & \rtimes \\
[\![\{C_1, C_2\}]\!] & \triangleq & \tilde{\mu} x.\langle \mu\alpha_1.\overline{\langle \mu\alpha_2.\overline{\langle x \| [\alpha_1, \alpha_2] \rangle} \| [\![C_2]\!] \rangle} \| [\![C_1]\!] \rangle \\
[\![\llcorner F \lrcorner]\!] & \triangleq & \tilde{\lambda}(\bullet, \blacksquare).[\![F]\!] \\
\\
[\![x \Rightarrow E]\!] & \triangleq & \langle \bullet \| \tilde{\mu} x.\overline{\langle E \| \blacksquare \rangle} \rangle \\
[\![\alpha \Leftarrow C]\!] & \triangleq & \langle \mu\alpha.\overline{\langle \bullet \| C \rangle} \| \blacksquare \rangle \\
\\
[\![() \Rightarrow E]\!] & \triangleq & \langle \bullet \| \tilde{\mu}_.\overline{\langle [\![E]\!] \| \blacksquare \rangle} \rangle \\
[\![\{\} \Leftarrow C]\!] & \triangleq & \langle \mu_.\overline{\langle \bullet \| [\![C]\!] \rangle} \| \blacksquare \rangle \\
\\
[\![(x_1, x_2) \Rightarrow E]\!] & \triangleq & \langle \bullet \| \tilde{\mu} x.\overline{\langle x \| \pi_1[\tilde{\mu} x_1.\overline{\langle x \| \pi_2[\tilde{\mu} x_2.\overline{\langle [\![E]\!] \| \blacksquare \rangle}] \rangle}] \rangle} \rangle \\
[\![\{\alpha_1, \alpha_2\} \Leftarrow C]\!] & \triangleq & \langle \mu\alpha.\overline{\langle i_1(\mu\alpha_1.\overline{\langle i_2(\mu\alpha_2.\overline{\langle \bullet \| [\![C]\!] \rangle}) \| \alpha \rangle}) \| \alpha \rangle} \| \blacksquare \rangle \\
\\
[\![\overline{E}]\!] & \triangleq & \langle [\![E]\!] \| \bullet \cdot \blacksquare \rangle \\
[\![\underline{C}]\!] & \triangleq & \langle \bullet - \blacksquare \| [\![C]\!] \rangle
\end{array}$$

Au niveau du typage, la traduction donne

$$
\begin{array}{lcl}
[A \to B]^* & \triangleq & A^* \to B^* \\
[B \leftarrow A]^* & \triangleq & A^* - B^* \\
(A \times B)^* & \triangleq & A^* \wedge B^* \\
(A + B)^* & \triangleq & A^* \vee B^* \\
unit^* & \triangleq & \top \\
null^* & \triangleq & \bot \\
X^* & \triangleq & X \\
\\
[\![\Gamma \vdash E : A; \Delta]\!] & \triangleq & \Gamma^* \vdash [\![E]\!] : A^*; \Delta^* \\
[\![\Gamma; C : A \vdash \Delta]\!] & \triangleq & \Gamma^*; [\![C]\!] : A^* \vdash \Delta^* \\
[\![F : (\Gamma; A \vdash B; \Delta)]\!] & \triangleq & [\![F]\!] : (\Gamma^*, \bullet : A^* \vdash \blacksquare : B^*; \Delta^*)
\end{array}
$$

Nous conjecturons que la composition de l'interprétation avec les transformations CPS des calculs $\mu_n\tilde{\mu}^{\to - \wedge_m \vee^1_m \top \bot}$ et $\mu\tilde{\mu}_v^{\to - \wedge^1_m \vee_m \top \bot}$ coïncident avec les sémantiques dénotationnelles spécifiques de l'appel par nom et de l'appel par valeur du λ-calcul symétrique, telles que données par Filinski [Fil89a, Fil89b].

Notons qu'il reste à vérifier si les traductions de (E_1, E_2) et $\{C_1, C_2\}$ ne pourraient pas en fait se simplifier en $([\![E_1]\!], [\![E_2]\!])$ et $[[\![C_1]\!], [\![C_2]\!]]$.

3.14.2 Le λ-calcul symétrique de Barbanera et Berardi

Le λ-calcul symétrique λ^{Sym} de Barbanera et Berardi [BB96] est un calcul typé non déterministe. On va montrer que pour un quotient « naturel » de la syntaxe du λ-calcul symétrique, on obtient un isomorphisme avec le calcul $LK_{\mu\tilde{\mu}}^{\vee_a \wedge_a \to_a -_a}$ non déterministe.

La syntaxe du calcul λ^{Sym} est

$$
\begin{array}{llcl}
\text{Termes} & t, u & ::= & x \mid (t, t) \mid \sigma_1(t) \mid \sigma_2(t) \mid \lambda x.c \\
\text{Commandes} & c & ::= & t \star t
\end{array}
$$

et les règles de réduction sont

$$
\begin{array}{llcll}
(\beta) & \lambda x.c \star t & \xrightarrow{h} & c[x \leftarrow t] & \\
(\beta^\bot) & t \star \lambda x.c & \xrightarrow{h} & c[x \leftarrow t] & \\
(\pi) & (t_1, t_2) \star \sigma_i(t) & \xrightarrow{h} & t_i \star t & \text{i=1,2} \\
(\pi^\bot) & \sigma_i(t) \star (t_1, t_2) & \xrightarrow{h} & t \star t_i & \text{i=1,2} \\
(\eta) & \lambda x.(t \star x) & \xrightarrow{h} & t \quad x \text{ non libre dans } t & \\
(\eta^\bot) & \lambda x.(x \star t) & \xrightarrow{h} & t \quad x \text{ non libre dans } t &
\end{array}
$$

Notons que Barbanera et Berardi donnent aussi des règles de simplification ($\lambda x.C[t \star x] \to t$ si C ne lie pas de variables dans t) que nous ne considérerons pas ici.

Les types du système de typage sont définis à partir de types atomiques positifs et de types atomiques négatifs que l'on désigne respectivement par les symboles $X, Y, \ldots$ et $X^\bot, Y^\bot, \ldots$ La syntaxe des types est

$$
A, B \quad ::= \quad X \mid X^\bot \mid A \vee B \mid A \wedge B
$$

On définit une opération de dualité sur les types par

$$
\begin{array}{lcl}
(X)^\bot & \triangleq & X^\bot \\
(X^\bot)^\bot & \triangleq & X \\
(A \vee B)^\bot & \triangleq & A^\bot \wedge B^\bot \\
(A \wedge B)^\bot & \triangleq & A^\bot \vee B^\bot
\end{array}
$$

On étend la notion de types positifs et négatifs des atomes aux types arbitraires :
- X est positif,
- $X^\bot$ est négatif,

– $A \wedge B$ et $A \vee B$ sont positifs si B est positif, sinon, ils sont négatifs
Les règles d'inférence du système de typage sont

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash c : \bot}{\Gamma \vdash \lambda x.c : A^{\perp}}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash (t,u) : A \wedge B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \sigma_1(t) : A \vee B} \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash \sigma_2(t) : A \vee B}$$

$$\frac{\Gamma \vdash t : A^{\perp} \quad \Gamma \vdash u : A}{\Gamma \vdash t \star u : \bot}$$

Clairement, le calcul λ^{Sym} est équipé de constructeurs additifs pour le produit. Si l'on essaie d'associer aux expressions de type positif un terme et aux expressions de type négatif un contexte d'évaluation, on s'aperçoit que les paires peuvent être des paires de termes, ou des paires de contextes d'évaluation, ou des paires hybrides, ce qui revient à considérer aussi les constructeurs additifs de la somme, de l'implication, et de la soustraction.

On montre alors que le calcul λ^{Sym} est en correspondance avec la version non-déterministe du calcul $LK_{\mu\tilde{\mu}}^{\vee_a \wedge_a \rightarrow_a -_a}$ dans lequel μ et $\tilde{\mu}$ sont tous deux identifiés avec le λ du λ^{Sym} et les réductions appel par nom et appel par valeur sont superposées.

On définit l'« aplatissement » suivant de $\mu\tilde{\mu}^{\vee_a \wedge_a \rightarrow_a -_a}$ vers le langage du calcul λ^{Sym} :

$$\begin{array}{lcllcl}
\psi(x) & \triangleq & x & \psi(\alpha) & \triangleq & x_\alpha \\
\psi((v_1, v_2)) & \triangleq & (\psi(v_1), \psi(v_2)) & \psi([e_1, e_2]) & \triangleq & (\psi(e_1), \psi(e_2)) \\
\psi(e \cdot v) & \triangleq & (\psi(e), \psi(v)) & \psi(v \cdot e) & \triangleq & (\psi(v), \psi(e)) \\
\psi(\iota_i(v)) & \triangleq & \sigma_i(\psi(v)) & \psi(\pi_i[e]) & \triangleq & \sigma_i(\psi(e)) \\
\psi(\iota_1(e)) & \triangleq & \sigma_1(\psi(e)) & \psi(\pi_1[v]) & \triangleq & \sigma_1(\psi(v)) \\
\psi(\mu\alpha.c) & \triangleq & \lambda x_\alpha.\psi(c) & \psi(\tilde{\mu}x.c) & \triangleq & \lambda x.\psi(c) \\
\psi\langle v \| e \rangle & \triangleq & \psi(v) \star \psi(e) & & &
\end{array}$$

Soit l'extension de ce morphisme aux types de $LK_{\mu\tilde{\mu}}^{\vee_a \wedge_a \rightarrow_a -_a}$.

$$\begin{array}{lcl}
\psi(X) & \triangleq & X \\
\psi(A \wedge_a B) & \triangleq & \psi(A) \wedge \psi(B) \\
\psi(B -_a A) & \triangleq & \psi(A)^{\perp} \wedge \psi(B) \\
\psi(A \vee_a B) & \triangleq & \psi(A) \vee \psi(B) \\
\psi(A \rightarrow_a B) & \triangleq & \psi(A)^{\perp} \vee \psi(B)
\end{array}$$

où l'on observe que $\psi(A)$ est positif par construction quelque soit A (l'apparente incohérence de la coïncidence de la notion de la positivité pour $A \wedge B$ et $A \vee B$ vient de l'intervertion des arguments de $B -_a A$ dans la traduction).

On vérifie que ψ est un isomorphisme entre les types de $LK_{\mu\tilde{\mu}}^{\vee_a \wedge_a \rightarrow_a -_a}$ (qui sont construits à partir des atomes avec $\vee_a$, $\wedge_a$, $\rightarrow_a$ et $-_a$) et les types positifs de λ^{Sym}.

On étend alors ψ aux contextes de typage Γ et Δ de manière compositionnelle et on vérifie que ψ envoie bien les jugements de $LK_{\mu\tilde{\mu}}^{\vee_a \wedge_a \rightarrow_a -_a}$ vers des jugements du calcul λ^{Sym}.

Proposition 35 $\left.\begin{array}{l} \Gamma \vdash v : A \,|\, \Delta \\ \Gamma \,|\, e : A \vdash \Delta \\ c : (\Gamma \vdash \Delta) \end{array}\right\}$ *implique* $\left\{\begin{array}{l} \psi(\Gamma), \psi(\Delta)^{\perp} \vdash \psi(v) : \psi(A) \\ \psi(\Gamma), \psi(\Delta)^{\perp} \vdash \psi(e) : \psi(A)^{\perp} \\ \psi(\Gamma), \psi(\Delta)^{\perp} \vdash \psi(c) : \bot \end{array}\right.$

On vérifie aussi que ψ respecte les règles de réduction.

Proposition 36 *Soit c une commande*
- *Si $c \to_\mu c'$ alors $\psi(c) \to_\beta \psi(c')$*
- *Si $c \to_{\tilde\mu} c'$ alors $\psi(c) \to_{\beta^\perp} \psi(c')$*
- *Si $c \to_{\wedge_a^i} c'$ ou $c \to_{-_a^i} c'$ alors $\psi(c) \to_\pi \psi(c')$*
- *Si $c \to_{\vee_a^i} c'$ ou $c \to_{\to_a^i} c'$ alors $\psi(c) \to_{\pi^\perp} \psi(c')$*
- *Si $c \to_{\eta_\mu} c'$ alors $\psi(c) \to_\eta \psi(c')$*
- *Si $c \to_{\eta_{\tilde\mu}} c'$ alors $\psi(c) \to_{\eta^\perp} \psi(c')$*

Et pareillement pour la réduction d'un terme ou d'un contexte d'évaluation.

La fonction ψ est donc un morphisme de $LK_{\mu\tilde\mu}^{\wedge_a\vee_a\to_a-_a}$ vers λ^{Sym}. C'est en fait même un morphisme de $\mu\tilde\mu^{\wedge_a\vee_a\to_a-_a}$ vers l'extension non typée de λ^{Sym}. Toutefois, nous aurons besoin des types pour contraindre ψ à être bijectif.

En fait, même en se restreignant au fragment typé, ψ n'atteint pas tous les termes du calcul λ^{Sym}. En particulier, dans la traduction $\psi(v) \star \psi(e)$ de $\langle v \| e\rangle$, on observe que $\psi(v)$ a un type positif et $\psi(e)$ un type négatif. Ainsi, pour espérer que ψ soit surjective, il faut restreindre le calcul λ^{Sym} aux cas des expressions dont les sous-expressions de la forme $t \star u$ vérifient t de type positif. Ceci revient en fait à considérer le calcul λ^{Sym} quotienté par la relation d'équivalence $\simeq$ engendrée par l'identification entre $t \star u$ et $u \star t$, ce qui, après tout, correspond à l'intention de ce calcul dont les règles duales compensent la séquentialisation arbitraire que la syntaxe impose. Ainsi, sur le quotient, les paires de règles duales coïncident si bien que les nouvelles règles, sur $\lambda_{\simeq}^{Sym}$, sont simplement (β), (π) et (η). Soit $\hat\psi$ la composition de ψ avec la projection de λ^{Sym} dans $\lambda_{\simeq}^{Sym}$. On a

Proposition 37 *Soit c une commande*
- *Si $c \to_\mu c'$ ou $c \to_{\tilde\mu} c'$ alors $\hat\psi(c) \to_\beta \hat\psi(c')$*
- *Si $c \to_{\wedge_a^i} c'$ ou $c \to_{-_a^i} c'$ ou $c \to_{\vee_a^i} c'$ ou $c \to_{\to_a^i} c'$ alors $\hat\psi(c) \to_\pi \hat\psi(c')$*
- *Si $c \to_{\eta_\mu} c'$ ou $c \to_{\eta_{\tilde\mu}} c'$ alors $\hat\psi(c) \to_\eta \hat\psi(c')$*

Et pareillement pour la réduction d'un terme ou d'un contexte d'évaluation.

On définit maintenant un morphisme ϕ que l'on prouvera être le morphisme réciproque de $\hat\psi$. On prend dans chaque classe d'équivalence de $\lambda_{\simeq}^{Sym}$ le représentant pour lequel chaque partie droite de $\star$ a un type positif. Le morphisme ϕ se décompose en une fonction ϕ_P qui envoie un terme de type positif P vers un terme de type $\psi^{-1}(P)$ dans $LK_{\mu\tilde\mu}^{\wedge_a\vee_a\to_a-_a}$, une fonction ϕ_N qui envoie un terme de type négatif N vers un contexte d'évaluation de type $\psi^{-1}(N^\perp)$ dans $LK_{\mu\tilde\mu}^{\wedge_a\vee_a\to_a-_a}$, et une fonction ϕ qui envoie les termes de type $\perp$ vers les commandes de $LK_{\mu\tilde\mu}^{\wedge_a\vee_a\to_a-_a}$.

$$\phi(t \star u) \;=\; \langle \phi_P(t) \| \phi_{P^\perp}(u)\rangle \qquad t \text{ de type } P$$

$$\begin{array}{llllll}
\phi_P(x) & \triangleq & x & \phi_N(x) & \triangleq & \alpha_x \\
\phi_{P_1\wedge P_2}(t_1,t_2) & \triangleq & (\phi_{P_1}(t_1), \phi_{P_2}(t_2)) & \phi_{N_1\vee N_2}(u_1,u_2) & \triangleq & [\phi_{N_1}(u_1)|\phi_{N_2}(u_2)] \\
\phi_{N\wedge P}(u,t) & \triangleq & \phi_N(u)\cdot\phi_P(t) & \phi_{P\vee N}(t,u) & \triangleq & \phi_P(t)\cdot\phi_N(u) \\
\phi_{P_1\vee P_2}(\sigma_i(t)) & \triangleq & \iota_i(\phi_{P_i}(t)) & \phi_{N_1\wedge N_2}(\sigma_i(u)) & \triangleq & \pi_i[\phi_{N_i}(u)] \\
\phi_{N\vee P}(\sigma_1(u)) & \triangleq & \iota_1(\phi_N(u)) & \phi_{P\wedge N}(\sigma_1(t)) & \triangleq & \pi_1[\phi_P(t)] \\
\phi_P(\lambda x.c) & \triangleq & \mu\alpha_x.\phi(c) & \phi_N(\lambda x.c) & \triangleq & \tilde\mu x.\phi(c)
\end{array}$$

On a alors

Proposition 38 *Les fonctions ϕ et ψ constituent un isomorphisme entre les calculs typés $\lambda_{\simeq}^{Sym}$ et $LK_{\mu\tilde\mu}^{\wedge_a\vee_a\to_a-_a}$. Plus précisément :*
- *ϕ_P et ψ constituent un isomorphisme entre les jugements de la forme $\Omega \vdash t : P$ et ceux de la forme $\Gamma \vdash v : \psi^{-1}(P) \,|\, \Delta$*
- *ϕ_N et ψ constituent un isomorphisme entre les jugements de la forme $\Omega \vdash t : N$ et ceux de la forme $\Gamma \,|\, e : \psi^{-1}(N^\perp) \vdash \Delta$*
- *ϕ et ψ constituent un isomorphisme entre les jugements de la forme $\Omega \vdash t : \perp$ et ceux de la forme $c : (\Gamma \vdash \Delta)$*

Il est difficile d'étendre la correspondance au cas d'une généralisation non typée de $\lambda^{Sym}_{\simeq}$. Par exemple, la classe d'équivalence de $\lambda y.(y \star y)$ n'est pas dans l'image de $\hat{\psi}$ puisque cela exige que y soit à la fois une variable de terme et une variable de contexte d'évaluation, ce qui n'est pas possible dans le calcul $\mu\tilde{\mu}$.

On pourrait imaginer étendre le calcul $\lambda^{Sym}_{\simeq}$ avec une règle d'η-conversion pour le produit additif, telle que

$$(\eta_{\wedge_a}) \qquad (\lambda x.(t \star \iota_1(x)), \lambda x.(t \star \iota_2(x))) \quad = \quad t \qquad\qquad x \text{ non libre dans } v\,.$$

Tout terme de la forme $y \star y$ serait alors équivalent à un terme de la forme

$$y \star (\lambda x_1.(y \star \iota_1(x_1)), \lambda x_2.(y \star \iota_2(x_2)))$$

pour lequel y est toujours du même côté du symbole $\star$. Mais l'image de ψ vérifie aussi l'invariant que l'argument de droite u d'une paire (t, u) à droite du symbole $\star$ est l'image d'un contexte d'évaluation et que l'argument u' d'un $\iota_2(u')$ à droite du symbole $\star$ est aussi un contexte d'évaluation. Cela impose que x_2 soit à la fois une variable de terme et une variable de contexte d'évaluation, ce qui n'est toujours pas possible.

3.14.3 Le calcul dual de Wadler

Le « calcul dual » de Wadler [Wad03] fait suite à la conception du calcul $\overline{\lambda}\mu\tilde{\mu}$ [CH00]. C'est un calcul typé dont la syntaxe est celle du système $\mu\tilde{\mu}^{\wedge_a\vee_a\neg}$, c'est-à-dire du $\mu\tilde{\mu}$ équipé des connecteurs produit et somme avec leurs constructeurs additifs, ainsi que du connecteur négation. Le calcul dual de Wadler est équipé de deux systèmes de réduction symétriques l'un de l'autre, le premier pour l'appel par nom et le second pour l'appel par valeur. La réduction des constructeurs est « forte » dans le sens où, en appel par valeur (resp. en appel par nom) les arguments des constructeurs de valeurs (resp. de contextes d'évaluation linéaires) qui sont des termes (resp. des contextes d'évaluation) doivent être évalués en constructeurs linéaires à leur tour avant de pouvoir interagir. En particulier, l'évaluation des paires (resp. des contexte d'analyse de cas) est asymétrique en appel par valeur (resp. en appel par nom) : dans la pratique la première composante des constructeurs binaires est évaluée en premier. Ceci est réalisé par des règles d'expansion similaires à la règle $(\rightarrow^{\mu}_{\eta})$ définie en section 3.2.8.

Nous conjecturons que le calcul dual de Wadler correspond aux extensions canoniques obtenues, selon le modèle du calcul $\overline{\lambda}_\eta\mu\tilde{\mu}_v$, par ajout des règles d'η-conversion $\eta^R_{\wedge_a}$, $\eta^L_{\wedge^1_a}$ (et de leurs versions duales pour la somme) aux versions appel par nom et appel par valeur du calcul $\mu\tilde{\mu}^{\wedge_a\vee_a\neg}$.

Par ailleurs, nous pouvons plonger le calcul dual de Wadler dans le calcul $\mu\tilde{\mu}^{\wedge_a\vee_a\neg}$ en interprétant les constructeurs d'expressions linéaires par leur expansion évaluant les arguments à l'avance (par exemple la paire (v_1, v_2) est interprétée par $\mu\alpha.\langle v_1 \| \tilde{\mu}x_1.\langle v_2 \| \tilde{\mu}x_2.\langle\overline{(x_1, x_2) \| \alpha}\rangle\rangle\rangle$. Nous conjecturons que le plongement obtenu est un morphisme à la fois pour l'appel par nom et pour l'appel par valeur, ce qui permet de voir le calcul dual de Wadler comme sous-ensemble d'un calcul non-déterministe dont les restrictions à l'appel par nom et à l'appel par valeur s'obtiennent par déterminisation d'une unique paire critique. Toutefois, ce morphisme ne respecte pas les formes normales.

3.15 Les limites de la structure $\mu\tilde{\mu}$

3.15.1 Coupures croisées

Lors de sa preuve de cohérence de l'arithmétique par élimination des coupures, Gentzen [Gen38] utilisa une forme d'élimination appelée coupures croisées qui entrelace les répétitions (les contractions) de part et d'autre de la coupure, d'une manière radicalement distincte de l'approche $\lambda\mu$-calcul (ou logique linéaire) qui donne priorité à un côté qui duplique l'autre avant de laisser l'autre dupliquer les valeurs sous-termes du premier côté.

Une méthode d'élimination similaire aux coupures croisées à été présentée dans [CH94]. Cela reste à intégrer au cadre du système $\mu\tilde{\mu}$.

3.15.2 Théorie des types dépendants

Dès qu'on en vient à la théorie des types, et plus spécialement à la théorie des types avec produit dépendant, une orientation des séquents se met en place pour gérer la dépendance et la dualité est mise à mal.

On peut par exemple étendre la règle de typage du produit dépendant de la section 3.8 en une règle de typage du produit dépendant de la théorie des types pour lequel le domaine de quantification est le langage des preuves lui-même

$$\Pi_L \quad \frac{\Gamma \,|\, e : B[x \leftarrow v] \vdash \Delta}{\Gamma \,;\, v \cdot e : \Pi x : A.B(x) \vdash \Delta} \qquad \Pi_R \quad \frac{\Gamma, x : A \vdash v : B(x) \,|\, \Delta}{\Gamma \vdash \lambda x.v : \Pi x : A.B(x) \,|\, \Delta}$$

avec la restriction que x ne figure pas dans les formules de Γ et Δ.

Un premier problème se pose : l'extension directe de la règle de réduction de l'implication au cas du produit dépendant

$$(\Pi_{wrong}) \quad \langle \lambda x.v \| v' \cdot e \rangle \quad \rightarrow \quad \langle v' \| \tilde{\mu} x.\overline{\langle v \| e \rangle} \rangle$$

n'est plus typable car v est de type dépendant $B(x)$ alors que e est de type $B(v')$. Dans le cas de l'appel par nom, on peut adopter la règle typable

$$(\Pi_n^\beta) \quad \langle \lambda x.v \| v' \cdot e \rangle \quad \rightarrow \quad \langle v[x \leftarrow v'] \| e \rangle$$

mais le cas de l'appel par valeur continue de poser problème.

On pourrait essayer de se rabattre vers la règle de réduction suivante

$$(\Pi) \quad \langle \lambda x.v \| v' \cdot e \rangle \quad \rightarrow \quad \langle \mu\alpha.\overline{\langle v' \| \tilde{\mu} x.\overline{\langle v \| \alpha \rangle} \rangle} \| e \rangle \qquad \alpha \text{ frais}$$

dans laquelle on gèlerait le rédex $\langle \mu\alpha. \ldots \| e \rangle$ tant que le sous-rédex $\langle v' \| \tilde{\mu} x.\overline{\langle v \| \alpha \rangle} \rangle$ n'est pas réduit.

Cette approche est vaine pour les cas où v' duplique son contexte d'évaluation car le $\mu\alpha$ se mettrait à lier des occurrences de α de types distincts (du style $B(V_1)$, $B(V_2)$, ... pour chaque sous-valeur de v' se retrouvant ultimement en interaction avec $\tilde{\mu} x.\langle v \| \alpha \rangle$). L'adéquation avec le type $B(v')$ de e est alors sans espoir.

Le cas intuitionniste (c'est-à-dire v' liant linéairement son contexte d'évaluation) reste sans doute gérable mais la solution la plus simple pour intégrer l'appel par valeur semble être de restreindre d'emblée la règle de formation des contextes applicatifs du produit au cas de l'application de valeur

$$\Pi_L^v \quad \frac{\Gamma \,|\, e : B[x \leftarrow V] \vdash \Delta}{\Gamma \,;\, V \cdot e : \Pi x : A.B(x) \vdash \Delta}$$

de telle sorte que la règle contractée

$$(\Pi_v^\beta) \quad \langle \lambda x.v \| V \cdot e \rangle \quad \rightarrow \quad \langle v[x \leftarrow V] \| e \rangle$$

soit typable.

Passons à l'étude de la soustraction dépendante (notons-la $\delta x : A.B(x)$). Cette fois, en raison de l'orientation de la dépendance de type de la gauche vers la droite, on se retrouve avec un jugement de typage de $\tilde{\lambda}\beta.e$ dont le type de β fait référence à un terme qui ne sera connu qu'au moment où l'expression $\tilde{\lambda}\beta.e$ sera intégrée à une commande. On représente cette dépendance par une variable spéciale $\bullet$, ce qui donne :

$$-_L \quad \frac{\Gamma \,|\, e : A \vdash \beta : B(\bullet), \Delta}{\Gamma \,;\, \tilde{\lambda}\beta.e : \delta x : A.B(x) \vdash \Delta} \qquad -_R \quad \frac{\Gamma \,|\, e : B[x \leftarrow v] \vdash \Delta \qquad \Gamma \vdash v : A \,|\, \Delta}{\Gamma \vdash v - e : \delta x : A.B(x) \,;\, \Delta}$$

avec la restriction que x ne figure pas dans les formules de Γ et Δ.

La règle de réduction

$$(\delta) \quad \langle v - e' \| \tilde{\lambda}\beta.e \rangle \quad \rightarrow \quad \langle \mu\beta.\overline{\langle v \| e \rangle} \| e' \rangle$$

est typable à condition de généraliser la règle de typage de la coupure au cas d'une dépendance du type du contexte d'évaluation envers le terme mis en interaction avec ce contexte :

$$Cut \quad \frac{\Gamma \vdash v : A \,|\, \Delta \quad \Gamma \,|\, e : A \vdash \Delta(\bullet)}{\langle v \| e \rangle : (\Gamma \vdash \Delta(v))}$$

Tant que le flux de calcul va du terme vers le contexte d'application, on peut espérer garder une certaine cohérence en présence de $\bullet$, mais si le flux s'inverse, alors, à moins d'être dans le fragment intuitionniste de telle sorte que les occurrences de $B(\bullet)$ ne soient pas dupliquées, on va droit vers une logique incohérente. C'est d'ailleurs ce qui a été montré dans [Her05] pour une théorie des types classique avec types Σ (les types Σ ne diffèrent calculatoirement de la soustraction dépendante que par la nature de l'objet de type dépendant : terme au lieu de contexte).

Il semble clair que la dualité trouve ainsi dans la théorie des types dépendants de fortes limites.

Chapitre 4

Opérateurs de liaison dynamique du contexte d'évaluation

L'étude théorique des opérateurs de contrôle et de leurs délimiteurs

La deuxième moitié des années 1980 vit une certaine effervescence autour de l'étude théorique et de la conception de nouveaux opérateurs de contrôle.

C'est l'époque où, motivés par le souci de développer un cadre formel de type λ-calcul pour étudier les opérateurs de continuations du type du `call-with-current-continuation` de Scheme, Felleisen, Duba et Kohlbecker, étudiants du groupe de Daniel Friedman à l'université d'Indiana, introduisirent l'opérateur $\mathcal{C}$ et le calcul $\lambda_{\mathcal{C}}$ qui allait, via Griffin, faire prendre conscience aux théoriciens de la démonstration que la logique classique était implémentable.

C'est aussi le moment où la notion de délimiteur de continuation vit le jour. Par analogie avec le rôle de la boucle interactive de Scheme, cet opérateur fut appelé *prompt* (notation #) et finalement décrit dans un article de 1988 [Fel88], conjointement avec un autre opérateur à la sémantique assez complexe, l'opérateur $\mathcal{F}$, qui finit par être finement analysé par Shan [Sha04].

De l'autre côté de l'Atlantique, c'est aussi l'effervescence, au Danemark, où Danvy et Filinski mettent au point un opérateur de composition de continuations dénommé `shift` [DF89], opérateur fonctionnant de pair avec un délimiteur de continuation nommé `reset` (notation $\langle\ \ \rangle$ de même nature que le *prompt* de Felleisen).

Contrairement à l'opérateur $\mathcal{F}$ et au délimiteur *prompt* dont la sémantique fut originellement décrite par un système de réduction, l'opérateur *shift* et le délimiteur *reset* furent initialement définis par leur comportement dans le langage cible de la traduction par passage de continuation de l'appel par valeur (« call-by-value continuation-passing-style », ou CPS CBV) de Fisher-Plotkin [Fis72, Fis93, Plo75].

Du $\lambda_{\mathcal{C}}$-calcul au $\lambda\mu$-calcul

Il existe deux variantes du calcul $\lambda_{\mathcal{C}}$. La première, de 1986 [FFKD86], mélangeait règles locales et règles applicables uniquement à la racine du terme. La seconde, de 1992 [FH92], s'affranchissait des règles applicables uniquement à la racine, mais en échange, afin de préserver la confluence, ne permettait pas, en appel par valeur, de réduire systématiquement les termes vers une valeur. En particulier, les expressions $\mathcal{C}(\lambda k.V)$ et $\mathcal{C}(\lambda k.k\,V)$ était des formes normales de tête qui ne pouvaient pas être réduites plus avant.

Les défauts de ce calcul compliquèrent la tâche de Griffin lorsqu'il s'agit de montrer que $\mathcal{C}$ était typable de type l'élimination de la double négation.

En comparaison, le calcul $\lambda\mu$ de Parigot [Par92] peut être vu comme une présentation « propre » du calcul $\lambda_{\mathcal{C}}$. Un des points important de ce succès est l'utilisation d'une substitution structurelle des contextes d'évaluation alors que le calcul $\lambda_{\mathcal{C}}$ repose sur une réification des contextes d'évaluation en continuations.

Une transcription du calcul $\lambda\mu$ dans la syntaxe du calcul $\lambda_{\mathcal{C}}$ fut présentée dans Ariola *et al* [AH03] et une analyse comparative des versions appel par valeur de la révision de 1992 de $\lambda_{\mathcal{C}}$ et de cette version de $\lambda_{\mathcal{C}}$ isomorphe au $\lambda\mu$-calcul est donnée dans Ariola *et al* [AFH05].

De *shift*/*reset* aux extensions monadiques

En 1994, Filinski [Fil94] explicita le fait que *reset* pouvait être identifié avec *prompt* et que *shift* pouvait se simuler à partir de $\mathcal{C}$ et *prompt* (ou, pour être plus précis, il explicita que la CPS CBV de $\lambda x.\mathcal{C}(\lambda k.x(\lambda y.\#(k\,y)))$ était celle définissant *shift*).

Dans le même article, Filinski prouva un résultat remarquable qui allait donner de la vigueur à l'étude de *shift* et *reset*. Il prouva que *shift* et *reset* ont exactement le pouvoir de représenter toutes les extensions monadiques exprimables en λ-calcul pur.

Dans le même article, Filinski proposait aussi une implantation de *shift* et *reset* à base de *callcc*, *abort* et une cellule mémoire (c'est-à-dire à base de $\mathcal{C}$ et une cellule mémoire, à ceci près qu'il n'existe pas d'implantation standard de $\mathcal{C}$ alors qu'il existe des implantations de *callcc* en Scheme et ML).

Le fait que *shift* et *reset* correspondent exactement à l'ajout d'une variable d'état à un calcul telle que $\lambda_{\mathcal{C}}$ ou le $\lambda\mu$-calcul, a été formalisé par Ariola *et al* [AHS04, AHS05b] qui donnent par ailleurs une décomposition fine, tant de *shift* que de $\mathcal{C}$ et de *reset* dans un calcul nommé $\lambda_{c\widehat{tp}}$.

Intuitivement, la monade de continuations s'obtient en ne gardant que $\mathcal{C}$ (le délimiteur est inutile) ; la monade d'exceptions s'obtient en restreignant $\mathcal{C}$ au cas d'un simple *abort* (c'est-à-dire un K-rédex) qui joue le rôle du lanceur d'exception (le « throw ») et en gardant le délimiteur qui joue alors le rôle de rattrapeur d'exception à liaison dynamique (le « handle »).

Dans le but de mieux comprendre la place des délimiteurs dans la dualité, mais aussi de comprendre si, à terme, les délimiteurs ont un rôle à jouer pour la complétude observationnelle du calcul $\overline{\lambda}\mu\tilde{\mu}$, nous donnons ci-après une reformulation du calcul $\lambda_{c\widehat{tp}}$ dans la syntaxe du calcul $\overline{\lambda}\mu\tilde{\mu}$.

4.1 Le calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$

Le calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ est une simple extension du calcul $\overline{\lambda}\mu\tilde{\mu}$ avec un lieur dynamique de contexte d'évaluation noté $\widehat{\mu}$. On se donne un nouvel ensemble de noms pour les variables dynamiques. On note ces noms avec des lettres minuscules grecques grasses, telles que $\hat{\alpha}$, $\hat{\beta}$, etc. La nouvelle construction $\widehat{\mu}\hat{\alpha}.c$, lorsque placée dans un contexte e, bloque la capture de e par $\hat{\alpha}$ et fait barrière entre l'évaluation de c et e. De fait, la liaison étant dynamique, on ne peut pas déterminer quelles occurrences de $\hat{\alpha}$ dans c vont être liées au $\widehat{\mu}\hat{\alpha}$ considéré tant que c ne sera pas en forme normale de tête (c'est-à-dire sous la forme $\langle x\|E\rangle$ ou $\langle V\|\hat{\alpha}\rangle$ ou $\langle V\|\alpha\rangle$). Ce n'est donc que lorsque c se réduira en $\langle V\|\hat{\alpha}\rangle$ que le délimiteur se débloquera pour contracter $\langle\widehat{\mu}\hat{\alpha}.\overline{\langle V\|\hat{\alpha}\rangle}\|e\rangle$ en $\langle V\|e\rangle$. Si c se réduit en revanche en $\langle V\|\hat{\beta}\rangle$, alors le lieur $\widehat{\mu}\hat{\alpha}$ ne liera rien et $\langle\widehat{\mu}\hat{\alpha}.\overline{\langle V\|\hat{\beta}\rangle}\|e\rangle$ se contractera en $\langle V\|\hat{\beta}\rangle$.

La construction $\widehat{\mu}\hat{\alpha}.c$ étant bloquante, elle ne capture pas son contexte et se comporte donc à ce niveau là comme une valeur. Toutefois, en appel par valeur, elle sera réduite en une valeur avant d'être substituée. Pour caractériser ce comportement double de $\widehat{\mu}\hat{\alpha}.c$ nous introduisons la catégorie des pré-valeurs que l'on représente par la lettre majuscule W. La syntaxe du calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ est :

$$
\begin{array}{lcl}
c & ::= & \langle v\|e\rangle \\
V & ::= & x \mid \lambda x.v \\
W & ::= & V \mid \widehat{\mu}\hat{\alpha}.c \\
v & ::= & W \mid \mu\alpha.c \\
e & ::= & E \mid \tilde{\mu}x.c \\
E & ::= & \alpha \mid v\cdot e \mid \hat{\alpha}
\end{array}
$$

On considère les expressions à α-conversion près des noms de variables statiques de contextes et de termes mais pas des noms de variables dynamiques. Les règles de réduction qui concernent la partie non dynamique sont inchangées

$$
\begin{array}{lrcl}
(\rightarrow) & \langle\lambda x.v_1\|v_2\cdot e\rangle & \xrightarrow{h} & \langle v_2\|\tilde{\mu}x.\overline{\langle v_1\|e\rangle}\rangle \\
(\mu) & \langle\mu\alpha.c\|e\rangle & \xrightarrow{h} & c[\alpha\leftarrow e] \\
(\tilde{\mu}) & \langle v\|\tilde{\mu}x.c\rangle & \xrightarrow{h} & c[x\leftarrow v]
\end{array}
$$

à ceci près que la substitution capture les variables dynamiques au passage de l'opérateur $\widehat{\mu}$.

En raison de la liaison dynamique de $\widehat{\mu}$, la réduction $\langle\widehat{\mu}\hat{\alpha}.c\|e\rangle \to c[\hat{\alpha} \leftarrow e]$ n'est pas correcte. Les deux formes bridées suivantes sont toutefois correctes :

$$\begin{array}{lll} (\widehat{\mu}^{var}) & \langle\widehat{\mu}\hat{\alpha}.c\|\hat{\alpha}\rangle & \stackrel{h}{\to} \; c \\ (\eta_{\widehat{\mu}}) & \widehat{\mu}\hat{\alpha}.\langle W\|\hat{\alpha}\rangle & \stackrel{h}{\to} \; W \end{array}$$

Ceci se ressent au niveau du typage. Le non engagement en faveur de l'appel par valeur ou l'appel par nom implique que toutes les variables de même nom susceptibles d'être liées dynamiquement sont de même type, d'un type qui est global à toute la dérivation.

Dans la suite, on note Ω un contexte de typage des noms de variables liées dynamiquement. Le système de typage suivant est un système de types simples pour $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$.

$$\frac{\Gamma \vdash v : A \,|\, \Delta; \Omega \qquad \Gamma \,|\, e : A \vdash \Delta; \Omega}{\langle v\|e\rangle : (\Gamma \vdash \Delta; \Omega)}$$

$$\frac{}{\Gamma\,;\,\alpha : A \vdash \alpha : A, \Delta; \Omega} \qquad \frac{}{\Gamma, x : A \vdash x : A\,;\, \Delta; \Omega}$$

$$\frac{\Gamma\,;\, E : A \vdash \Delta; \Omega}{\Gamma \,|\, E : A \vdash \Delta; \Omega} \qquad \frac{\Gamma \vdash V : A\,;\, \Delta; \Omega}{\Gamma \vdash V : A \,|\, \Delta; \Omega}$$

$$\frac{c : (\Gamma, x : A \vdash \Delta; \Omega)}{\Gamma \,|\, \tilde{\mu}x.c : A \vdash \Delta; \Omega} \qquad \frac{c : (\Gamma \vdash \alpha : B, \Delta; \Omega)}{\Gamma \vdash \mu\alpha.c : B \,|\, \Delta; \Omega}$$

$$\frac{}{\Gamma \,|\, \hat{\alpha} : A \vdash \Delta; \hat{\alpha} : A, \Omega} \qquad \frac{c : (\Gamma \vdash \Delta; \hat{\alpha} : A, \Omega)}{\Gamma \vdash \widehat{\mu}\hat{\alpha}.c : A \,|\, \Delta; \hat{\alpha} : A, \Omega}$$

$$\frac{\Gamma \vdash v : A \,|\, \Delta; \Omega \qquad \Gamma \,|\, e : B \vdash \Delta; \Omega}{\Gamma\,;\, v \cdot e : A \to B \vdash \Delta; \Omega} \qquad \frac{\Gamma, x : A \vdash v : B \,|\, \Delta; \Omega}{\Gamma \vdash \lambda x.v : A \to B\,;\, \Delta; \Omega}$$

Proposition 39 *Le typage est préservé par réduction.*

4.2 Le calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ par valeur

Danvy et Filinski montrèrent qu'on pouvait donner un typage plus précis à base de types simples pour le λ-calcul avec *shift* et *reset* CBV à condition d'annoter les implications avec des « effets ».

On peut adapter leur système au calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ CBV comme suit :

$$A, B, C, D \;::=\; X \,\|\, A \;{}_{\Gamma}{\to}_{\Gamma}\; A$$

$$\frac{\Gamma; \Omega' \vdash v : A \,|\, \Delta; \Omega \qquad \Gamma \,|\, e : A \vdash \Delta; \Omega'}{\langle v\|e\rangle : (\Gamma \vdash \Delta; \Omega)}$$

$$\frac{}{\Gamma \,|\, \alpha : A \vdash \alpha : A_{\Omega}, \Delta; \Omega} \qquad \frac{\dfrac{}{\Gamma, x : A \vdash x : A\,;\, \Delta}}{}$$

$$\frac{\Gamma \vdash V : A\,;\, \Delta}{\Gamma; \Omega \vdash V : A \,|\, \Delta; \Omega}$$

$$\frac{c : (\Gamma, x : A \vdash \Delta; \Omega)}{\Gamma \,|\, \tilde{\mu}x.c : A \vdash \Delta; \Omega} \qquad \frac{c : (\Gamma \vdash \alpha : B_{\Omega'}, \Delta; \Omega)}{\Gamma; \Omega' \vdash \mu\alpha.c : B \,|\, \Delta; \Omega}$$

$$\frac{\text{si } \hat{\alpha} : A \text{ est dans } \Omega}{\Gamma \,|\, \hat{\alpha} : A \vdash \Delta; \Omega} \qquad \frac{c : (\Gamma \vdash \Delta; \hat{\alpha} : A, \Omega)}{\Gamma; \hat{\alpha} : B, \Omega \vdash \widehat{\mu}\hat{\alpha}.c : A \,|\, \Delta; \hat{\alpha} : B, \Omega}$$

$$\frac{\Gamma;\Omega'' \vdash v : A\,;\,\Delta;\Omega \qquad \Gamma\,|\,e : B \vdash \Delta;\Omega'}{\Gamma\,|\,v\cdot e : A \;{}_{\Omega'}\!\!\to_{\Omega''} B \vdash \Delta;\Omega} \qquad \frac{\Gamma, x : A;\Omega' \vdash v : B\,|\,\Delta;\Omega}{\Gamma \vdash \lambda x.v : A\;{}_{\Omega'}\!\!\to_{\Omega} B\,;\,\Delta}$$

Le calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ en appel par valeur peut être plongé dans le calcul $\overline{\lambda}\mu\tilde{\mu}$ étendu avec la soustraction.

La traduction est donnée ci-dessous. Soit $(\hat{\alpha}_i)_{1\leq i\leq n}$ une énumération finie de variables dynamiques. Soit $\vec{e}$ est une suite finie de contextes d'évaluation en correspondance avec les $(\hat{\alpha}_i)_{1\leq i\leq n}$. La traduction s'applique aux expressions dont les variables dynamiques sont incluses dans $(\hat{\alpha}_i)_{1\leq i\leq n}$. On note $\tilde{\lambda}\vec{\alpha}.e$ pour $\tilde{\lambda}\vec{\alpha_1}.\dots\tilde{\lambda}\vec{\alpha_n}.e$.

$$\begin{array}{lcll}
[\![\langle V\|e\rangle]\!]_{\vec{e}} & \triangleq & \langle [\![V]\!]\|[\![e]\!]_{\vec{e}}\rangle & \\
[\![\langle \widehat{\mu}\hat{\alpha}_i.c\|e'_i\rangle]\!]_{\vec{e}} & \triangleq & [\![c]\!]_{e_1\dots e'_i\dots e_n} & \\
[\![\langle v\|e\rangle]\!]_{\vec{e}} & \triangleq & \langle [\![v]\!]_{\vec{e}}\|[\![e]\!]\rangle & \text{si } v \text{ de la forme } \mu\beta.c \\
 & & & \\
[\![x]\!] & \triangleq & x & \\
[\![\lambda x.v]\!] & \triangleq & \lambda x.\mu\beta.\langle x\|\tilde{\lambda}\vec{\alpha}.\tilde{\mu}x.\langle \overline{[\![v]\!]_{\vec{\alpha}}\|\beta}\rangle\rangle & \\
 & & & \\
[\![V]\!]_{\vec{e}} & \triangleq & [\![V]\!] - e_1\dots e_n & \\
[\![\widehat{\mu}\hat{\alpha}_i.c]\!] & \triangleq & \mu\alpha_i.[\![c]\!]_{e_1\dots e_{i-1}\alpha_i e_{i+1}\dots e_n} - e_1\dots e_n & \\
[\![\mu\beta.c]\!]_{\vec{e}} & \triangleq & \mu\beta.[\![c]\!]_{\vec{e}} & \\
 & & & \\
[\![e]\!] & \triangleq & \tilde{\lambda}\vec{\alpha}.[\![e]\!]_{\vec{\alpha}} & \\
 & & & \\
[\![\beta]\!]_{\vec{e}} & \triangleq & \beta & \\
[\![v\cdot e]\!]_{\vec{e}} & \triangleq & [\![v]\!]_{\vec{e}}\cdot[\![e]\!] & \\
[\![\tilde{\mu}x.c]\!]_{\vec{e}} & \triangleq & \tilde{\mu}x.[\![c]\!]_{\vec{e}} & \\
[\![\hat{\alpha}_i]\!]_{\vec{e}} & \triangleq & e_i &
\end{array}$$

où les $\vec{\alpha}$ sont un lot de variables fraîches de contexte de longueur n.

La traduction s'étend au système de types en appel par valeur comme suit :

$$\begin{array}{lcl}
X^* & \triangleq & X \\
(A\;{}_{\Omega'}\!\!\to_{\Omega} B)^* & \triangleq & (A^* - \Omega^*) \to (B^* - \Omega'^*) \\
 & & \\
(\epsilon)^* & \triangleq & \epsilon \qquad \text{(contexte vide)} \\
(\Gamma, x : A)^* & \triangleq & \Gamma^*, x : A^* \\
(\Delta, \alpha : A_\Omega)^* & \triangleq & \Delta^*, \alpha : A^* - \Omega^* \\
 & & \\
[\![\Gamma \vdash V : A\,;\,\Delta]\!] & \triangleq & \Gamma^* \vdash [\![V]\!] : A^*\,;\,\Delta^* \\
[\![\Gamma;\vec{U} \vdash v : A\,|\,\Delta;\vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{e}} & \triangleq & \Gamma^* \vdash [\![v]\!]_{\vec{e}} : (A^* - \vec{U}^*)\,|\,\Delta^* \\
[\![\Gamma\,|\,e : B \vdash \Delta;\vec{\hat{\alpha}} : \vec{T}]\!] & \triangleq & \Gamma^*\,|\,[\![e]\!] : (B^* - \vec{T}^*) \vdash \Delta^* \\
[\![c : (\Gamma \vdash \Delta;\vec{\hat{\alpha}} : \vec{T})]\!]_{\vec{e}} & \triangleq & [\![c]\!]_{\vec{e}} : (\Gamma^* \vdash \Delta^*)
\end{array}$$

où l'on a supposé $\Gamma^*\,|\,e_i : T_i^* \vdash \Delta^*$.

Proposition 40 (Typage de la traduction) *Soit $\vec{e}$ une substitution des variables dynamiques telle que $\Gamma^*\,|\,e_i : T_i^* \vdash \Delta^*$. On a*

$$\left.\begin{array}{l}
\Gamma \vdash V : A\,;\,\Delta \\
\Gamma;\vec{U} \vdash v : A\,|\,\Delta;\vec{\hat{\alpha}} : \vec{T} \\
\Gamma\,|\,e : B \vdash \Delta;\vec{\hat{\alpha}} : \vec{T} \\
c : (\Gamma \vdash \Delta;\vec{\hat{\alpha}} : \vec{T})
\end{array}\right\}\ \textit{implique}\ \left\{\begin{array}{l}
[\![\Gamma \vdash V : A\,;\,\Delta]\!] \\
[\![\Gamma;\vec{U} \vdash v : A\,|\,\Delta;\vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{e}} \\
[\![\Gamma\,|\,e : B \vdash \Delta;\vec{\hat{\alpha}} : \vec{T}]\!] \\
[\![c : (\Gamma \vdash \Delta;\vec{\hat{\alpha}} : \vec{T})]\!]_{\vec{e}}
\end{array}\right.$$

En vue de l'énoncé de la complétude du système de réduction vis à vis de son plongement dans le calcul avec soustraction, il convient d'ajouter la règle suivante :

$$(\widehat{\mu}^{\text{lift}}) \qquad \langle \widehat{\mu}\hat{\alpha}.\langle \overline{\widehat{\mu}\hat{\alpha}.c\|\tilde{\mu}x.c'}\rangle\|e\rangle \quad \xrightarrow{h}_v \quad \langle \widehat{\mu}\hat{\alpha}.c\|\tilde{\mu}x.\langle \overline{\widehat{\mu}\hat{\alpha}.c'\|e}\rangle\rangle$$

dans laquelle il peut être opportun de préciser que les deux occurrences liantes de $\hat{\alpha}$ doivent bien être de même nom.

On peut alors faire une comparaison avec l'axiomatique du calcul $\lambda_{\mathcal{S}}$ (λ-calcul étendu avec *shift* et *reset*) de Kameyama et Hasegawa [KH03]. La comparaison avec cette axiomatique, complète vis à vis de la transformation CPS du calcul $\lambda_{\mathcal{S}}$, est présentée en section 4.3. La complétude du calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}_v$ vis à vis de son plongement dans le calcul $\overline{\lambda}\mu\tilde{\mu}_v$ avec soustraction (conjecturée ci-dessous) suggère donc que le calcul $\overline{\lambda}\mu\tilde{\mu}_v$ avec soustraction est lui-même complet vis à vis de sa transformation CPS.

Proposition 41 (Validité de la traduction) *Si $t \to t'$ dans $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}_v$, alors $[\![t]\!]_{\vec{e}} \xrightarrow{*} [\![t']\!]_{\vec{e}}$ dans $\overline{\lambda}\mu\tilde{\mu}_v$ avec soustraction.*

Proposition 42 (Préservation des formes normales) *Si v est en forme normale alors $[\![v]\!]_{\alpha_1 \ldots \alpha_n}$ l'est aussi.*

Conjecture 1 (Complétude de la traduction) *Si, pour tout $\vec{e}$, $[\![t]\!]_{\vec{e}} \to [\![t']\!]_{\vec{e}}$ dans $\overline{\lambda}\mu\tilde{\mu}_v$ avec soustraction, alors $t \xrightarrow{*} t'$ dans $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}_v$.*

4.3 Le calcul $\overline{\lambda}\mu\tilde{\mu}\#$ par valeur

On se restreint maintenant à une unique variable dynamique de contexte d'évaluation que l'on note tp dans la lignée de Ariola *et al* [AHS04, AHS05b]. On adopte pareillement la notation $\#c$ pour désigner $\widehat{\mu}\mathsf{tp}.c$ de telle sorte que l'aspect dynamique de la variable tp devient implicite. La version CBV du calcul obtenu, que l'on appelle $\overline{\lambda}\mu\tilde{\mu}_v\#$ est décrite par la syntaxe :

$$\begin{array}{lcl} c & ::= & \langle v \| e \rangle \\ V & ::= & x \mid \lambda x.v \\ W & ::= & V \mid \#c \\ v & ::= & W \mid \mu\alpha.c \\ e & ::= & \tilde{\mu}x.c \mid \alpha \mid v \cdot e \mid \mathsf{tp} \end{array}$$

et les règles de réduction ou conversion

$$\begin{array}{llcll} (\to) & \langle \lambda x.v_1 \| v_2 \cdot e \rangle & \xrightarrow{h}_v & \langle v_2 \| \tilde{\mu}x.\langle \overline{v_1 \| e} \rangle \rangle & \\ (\mu) & \langle \mu\alpha.c \| e \rangle & \xrightarrow{h}_v & c[\alpha \leftarrow e] & \\ (\tilde{\mu}_v) & \langle V \| \tilde{\mu}x.c \rangle & \xrightarrow{h}_v & c[x \leftarrow V] & \\ (\#) & \langle \#c \| \mathsf{tp} \rangle & \xrightarrow{h}_v & c & \\ & & & & \\ (\#^{\mathrm{lift}}) & \langle \#\langle \overline{\#c \| \tilde{\mu}x.c'} \rangle \| e \rangle & \xrightarrow{h}_v & \langle \#c \| \tilde{\mu}x.\langle \overline{\#c' \| e} \rangle \rangle & \\ & & & & \\ (\eta^R_{\to v}) & \lambda x.\mu\alpha \langle V \| x \cdot \alpha \rangle & = & V & x \text{ non libre dans } V \\ (\eta_\mu) & \mu\alpha.\langle v \| \alpha \rangle & = & v & \alpha \text{ non libre dans } v \\ (\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x \| e \rangle & = & e & x \text{ non libre dans } e \\ (\eta_\#) & \#\langle W \| \mathsf{tp} \rangle & = & W & \end{array}$$

En exprimant ces règles dans une syntaxe proche de celle du $\lambda_{\mathcal{C}}$-calcul, on peut établir une correspondance informelle avec l'adaptation au $\lambda_{\mathcal{C}}$ des règles du calcul $\lambda_{\mathcal{S}}$ de Kameyama et Hasegawa [KH03] (on a remplacé les règles sur *shift* par leurs versions équivalentes sur $\mathcal{C}$) données sur la colonne de droite :

$(\to)$	$e[(\lambda x.v_1)v_2]$	$\xrightarrow{h}_v$	$\mathbf{let}\ x = v_2\ \mathbf{in}\ e[v_1]$	partie générique « β » de β_v
(μ)	$e[\mathcal{C}(\lambda\alpha.c)]$	$\xrightarrow{h}_v$	$c[\alpha \leftarrow e]$	$\mathcal{C}$-lift
$(\tilde{\mu}_v)$	$\mathbf{let}\ x = V\ \mathbf{in}\ c$	$\xrightarrow{h}_v$	$c[x \leftarrow V]$	partie spécifique « $_v$ » de β_v
$(\#)$	$\mathsf{tp}\,(\#c)$	$\xrightarrow{h}_v$	c	implicite dans $\mathcal{C}$-reset
$(\#^{\text{lift}})$	$e[\#\mathbf{let}\ x = \#c\ \mathbf{in}\ c']$	$\xrightarrow{h}_v$	$\mathbf{let}\ x = \#c\ \mathbf{in}\ e[\#c']$	reset-lift
$(\eta^R_{\to v})$	$\lambda x.\mathcal{C}(\lambda\alpha.\alpha\,(V\,x))$	$=$	V	$\eta_v + \mathcal{C}$-elim (x et α non libres dans V)
(η_μ)	$\mathcal{C}(\lambda\alpha.\alpha\,v)$	$=$	v	$\mathcal{C}$-elim (x et α non libres dans v)
$(\eta_{\tilde{\mu}})$	$\mathbf{let}\ x = e[\,]\ \mathbf{in}\ x$	$=$	$e[\,]$	β_Ω (x non libre dans e)
$(\eta_\#)$	$\#\,(\mathsf{tp}\,W)$	$=$	W	généralisation de reset-value

Enfin, en se basant sur [AHS05b], les équivalents de *shift* et *reset* dans ce calcul devraient être définissables par

$$\begin{array}{lcll} \textit{shift}\ v & \triangleq & \mu\alpha.\langle v \| (\lambda x.\#\langle \overline{x \| \alpha} \rangle) \cdot \mathsf{tp} \rangle & \quad \text{pour } x \text{ et } \alpha \text{ frais} \\ \textit{reset}\ v & \triangleq & \#\langle v \| \mathsf{tp} \rangle & \end{array}$$

4.4 Le calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ par nom

On peut aussi donner une version appel par nom qui transfère l'état des variables du contexte d'évaluation vers le terme.

$$\frac{\Gamma \vdash v : A \,|\, \Delta; \Omega' \qquad \Gamma; \Omega' \,|\, e : A \vdash \Delta; \Omega}{\langle v \| e \rangle : (\Gamma \vdash \Delta; \Omega)}$$

$$\frac{}{\Gamma\,;\ \alpha : A \vdash \alpha : A, \Delta; \Omega} \qquad \frac{}{\Gamma, x : A_\Omega \vdash x : A \,|\, \Delta; \Omega}$$

$$\frac{\Gamma\,;\ E : A \vdash \Delta; \Omega}{\Gamma; \Omega' \,|\, E : A \vdash \Delta; \Omega}$$

$$\frac{c : (\Gamma, x : A_{\Omega'} \vdash \Delta; \Omega)}{\Gamma; \Omega' \,|\, \tilde{\mu}x.c : A \vdash \Delta; \Omega} \qquad \frac{c : (\Gamma \vdash \alpha : A, \Delta; \Omega)}{\Gamma \vdash \mu\alpha.c : A \,|\, \Delta; \Omega}$$

$$\frac{\text{si } \hat{\alpha} : A \text{ est dans } \Omega}{\Gamma\,;\ \hat{\alpha} : A \vdash \Delta; \Omega} \qquad \frac{c : (\Gamma \vdash \Delta; \hat{\alpha} : A, \Omega)}{\Gamma \vdash \widehat{\mu}\hat{\alpha}.c : A \,|\, \Delta; \hat{\alpha} : B, \Omega}$$

$$\frac{\Gamma \vdash v : A \,|\, \Delta; \Omega' \qquad \Gamma; \Omega'' \,|\, e : B \vdash \Delta; \Omega}{\Gamma\,;\ v \cdot e : A \;_{\Omega' \to \Omega''}\; B \vdash \Delta; \Omega} \qquad \frac{\Gamma, x : A_{\Omega'} \vdash v : B \,|\, \Delta; \Omega''}{\Gamma \vdash \lambda x.v : A \;_{\Omega' \to \Omega''}\; B \,|\, \Delta; \Omega}$$

Le calcul $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}$ en appel par nom peut aussi être plongé dans un calcul $\overline{\lambda}\mu\tilde{\mu}$ sans variable dynamique. Cette fois, on utilise la somme avec un constructeur de valeur multiplicatif asymétrique (cf section 3.6.2).

La traduction est donnée ci-dessous. Soit $(\hat{\alpha}_i)_{1 \le i \le n}$ une énumération finie de variables dynamiques. Soit $\vec{E}$ une suite finie de contextes d'évaluation linéaires en correspondance avec les $(\hat{\alpha}_i)_{1 \le i \le n}$. La traduction s'applique aux expressions dont les variables dynamiques sont incluses dans $(\hat{\alpha}_i)_{1 \le i \le n}$. On note $\lambda_1\vec{\alpha}.v$ pour $\lambda_1\alpha_1.\ldots\lambda_1\alpha_n.v$ et $[\vec{E}, E']$ pour $[E_1, \ldots [E_n, E']\ldots]$.

$$
\begin{array}{lcll}
[\![\langle v \| \tilde{\mu}x.c\rangle]\!]_{\vec{E}} & \triangleq & \langle [\![v]\!] \| [\![\tilde{\mu}x.c]\!]_{\vec{E}}\rangle & \\
[\![\langle v \| E\rangle]\!]_{\vec{E}} & \triangleq & \langle [\![v]\!]_{\vec{E}} \| [\![E]\!]'_{\vec{E}}\rangle & \text{sinon} \\
 & & & \\
[\![v]\!] & \triangleq & \lambda_1 \vec{\alpha}.[\![v]\!]_{\vec{\alpha}} & \\
 & & & \\
[\![x]\!]_{\vec{E}} & \triangleq & \mu\beta.\langle x \| [\vec{E}, \beta]\rangle & \\
[\![\lambda x.v]\!]_{\vec{E}} & \triangleq & \lambda x.[\![v]\!] & \\
[\![\widehat{\mu}\hat{\alpha}_i.c]\!]_{\vec{E}} & \triangleq & \mu\alpha_i.[\![c]\!]_{E_1 \ldots E_{i-1}\alpha_i E_{i+1} E_n} & \\
[\![\mu\beta.c]\!]_{\vec{E}} & \triangleq & \mu\beta.[\![c]\!]_{\vec{E}} & \\
 & & & \\
[\![\tilde{\mu}x.c]\!]_{\vec{E}} & \triangleq & \tilde{\mu}x.[\![c]\!]_{\vec{E}} & \\
[\![E]\!]_{\vec{E}} & \triangleq & [\vec{E}, [\![E]\!]'_{\vec{E}}] & \\
 & & & \\
[\![v \cdot e]\!]'_{\vec{E}} & \triangleq & [\![v]\!] \cdot [\![e]\!]_{\vec{E}} & \\
[\![\beta]\!]'_{\vec{E}} & \triangleq & \beta & \\
[\![\hat{\alpha}_i]\!]'_{\vec{E}} & \triangleq & E_i &
\end{array}
$$

où les $\vec{\alpha}$ sont un lot de variables fraîches de contexte de longueur n (en particulier, dans $[\![\widehat{\mu}\hat{\alpha}_i.c]\!]$, l'occurrence liante de α_i dans $\lambda_1\vec{\alpha}.$ ne lie aucune variable car l'occurrence liante de α_i dans $\mu\alpha_i.$ prend précédence).

La traduction s'étend au système de types en appel par nom comme suit :

$$
\begin{array}{lcl}
X^* & \triangleq & X \\
(A \;{}_{\Omega'}{\rightarrow}_{\Omega}\; B)^* & \triangleq & (\Omega^* \vee A^*) \rightarrow (\Omega'^* \vee B^*) \\
 & & \\
(\epsilon)^* & \triangleq & \epsilon \qquad \text{(contexte vide)} \\
(\Gamma, x : A_\Omega)^* & \triangleq & \Gamma^*, x : \Omega^* \vee A^* \\
(\Delta, \alpha : A)^* & \triangleq & \Delta^*, \alpha : A^* \\
 & & \\
[\![\Gamma \vdash v : A \,|\, \Delta; \vec{\hat{\alpha}} : \vec{T}]\!] & \triangleq & \Gamma^* \vdash [\![v]\!] : (\vec{T}^* \vee A^*) \,|\, \Delta^* \\
[\![\Gamma \vdash v : A \,|\, \Delta; \vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{E}} & \triangleq & \Gamma^* \vdash [\![v]\!]_{\vec{E}} : A^* \,|\, \Delta^* \\
[\![\Gamma; \vec{U} \,|\, e : A \vdash \Delta^*; \vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{E}} & \triangleq & \Gamma \,|\, [\![e]\!]_{\vec{E}} : (\vec{U}^* \vee A^*) \vdash \Delta^* \\
[\![\Gamma\,;\, E : A \vdash \Delta^*; \vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{E}} & \triangleq & \Gamma\,;\, [\![E]\!]'_{\vec{E}} : A^* \vdash \Delta^* \\
[\![c : (\Gamma \vdash \Delta; \vec{\hat{\alpha}} : \vec{T})]\!]_{\vec{E}} & \triangleq & [\![c]\!]_{\vec{E}} : (\Gamma^* \vdash \Delta^*)
\end{array}
$$

où l'on a supposé $\Gamma^*\,;\, E_i : T_i^* \vdash \Delta^*$.

Proposition 43 (Typage de la traduction) *Soit $\vec{E}$ une substitution des variables dynamiques telle que $\Gamma^* \,|\, E_i : T_i^* \vdash \Delta^*$. On a*

$$
\left.\begin{array}{l}
\Gamma \vdash v : A \,|\, \Delta; \vec{\hat{\alpha}} : \vec{T} \\
\Gamma\,;\, E : A \vdash \Delta \\
\Gamma; \vec{U} \,|\, e : A \vdash \Delta; \vec{\hat{\alpha}} : \vec{T} \\
c : (\Gamma \vdash \Delta; \vec{\hat{\alpha}} : \vec{T})
\end{array}\right\} \textit{implique} \left\{\begin{array}{l}
[\![\Gamma \vdash v : A \,|\, \Delta; \vec{\hat{\alpha}} : \vec{T}]\!] \textit{ et } [\![\Gamma \vdash v : A \,|\, \Delta; \vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{E}} \\
[\![\Gamma\,;\, E : A \vdash \Delta]\!]_{\vec{E}} \\
[\![\Gamma; \vec{U} \,|\, e : A \vdash \Delta; \vec{\hat{\alpha}} : \vec{T}]\!]_{\vec{E}} \\
[\![c : (\Gamma \vdash \Delta; \vec{\hat{\alpha}} : \vec{T})]\!]_{\vec{E}}
\end{array}\right.
$$

Il reste à montrer que la traduction est bien un morphisme pour la réduction de $\overline{\lambda}\mu\tilde{\mu}\widehat{\mu}_n$ vers $\overline{\lambda}\mu\tilde{\mu}_n$ avec somme.

Conclusion

Au delà de l'explication que le calcul $\mu\tilde{\mu}$ donne de la symétrie du calcul des séquents en termes d'une symétrie entre appel par nom et appel par valeur et entre termes et contextes d'évaluation, et au delà de l'impact de cette symétrie sur l'explicitation syntaxique d'une dualité entre appel par nom et appel par valeur, on peut dire du calcul $\mu\tilde{\mu}$ qu'il se présente aussi comme un bon outil d'analyse des diverses notions calculatoires, explicitant des dualités jusque là implicites telles que la dualité récursion/corécursion, révélant des dualités inconnues telles que la dualité entre appel par valeur paresseux et appel par nom paresseux, et posant les limites mêmes de la dualité, par exemple en présence de types dépendants dans le style de la théorie des types.

Ce mémoire peut être vu comme l'ébauche d'un projet de refonder au dessus du système $\mu\tilde{\mu}$ une théorie du calcul traitant uniformément de l'appel par nom, de l'appel par valeur et des calculs intuituitionniste et classique. L'analyse de l'η-réduction et l'ouverture vers les délimiteurs de continuations peuvent par exemple être vues comme des étapes vers la formulation d'un théorème de Böhm pour les extensions du système $\mu\tilde{\mu}$. Dans un mouvement de fertilisation inverse, ce mémoire suggère aussi que la simplicité avec laquelle l'appel par valeur se caractérise dans le système $\mu\tilde{\mu}$ peut déboucher vers de meilleurs outils d'analyse de la théorie du λ-calcul en appel par valeur, théorie qui, en comparaison de celle du λ-calcul usuel (c'est-à-dire en appel par nom), reste sensiblement sous-développée.

Remerciements

Ce mémoire a été rédigé à l'occasion de mon passage d'habilitation à diriger les recherches. Ce fut un long processus de *passage* (neuf mois !) et mes remerciements vont tout spécialement à Jean-Pierre Jouannaud et Christine Paulin pour leur soutien, ainsi qu'aux trois rapporteurs et trois examinateurs dont le nom figure en couverture et qui m'ont fait l'honneur, et le plaisir, de bien vouloir s'intéresser à mes travaux.

Concernant ce mémoire à proprement parler, je remercie particulièrement Pierre-Louis Curien et Andrzej Filinski pour leur relecture soigneuse.

Bibliographie

[AF93] Zena ARIOLA et Matthias FELLEISEN. « The Call-By-Need Lambda Calculus ». *J. Funct. Program.*, 7(3) :265–301, 1993.

[AFH05] Zena M. ARIOLA, Matthias FELLEISEN et Hugo HERBELIN. « Control Reduction Theories ». Soumis, 2005.

[AH03] Zena M. ARIOLA et Hugo HERBELIN. « Minimal Classical Logic and Control Operators ». Dans *Thirtieth International Colloquium on Automata, Languages and Programming , ICALP'03, Eindhoven, The Netherlands, June 30 - July 4, 2003*, volume 2719, pages 871–885. Springer-Verlag, LNCS, 2003.

[AHS04] Zena M. ARIOLA, Hugo HERBELIN et Amr SABRY. « A Type-Theoretic Foundation of Continuations and Prompts ». Dans *Proceedings of the Ninth ACM SIGPLAN International Conference on Functional Programming, ICFP 2004, Snowbird, UT, USA, September 19-21, 2004*, pages 40–53. ACM Press, New York, 2004.

[AHS05a] Zena M. ARIOLA, Hugo HERBELIN et Amr SABRY. « A Proof-Theoretic Foundation of Abortive Continuations ». *Higher Order and Symbolic Computation*, 2005. À paraître.

[AHS05b] Zena M. ARIOLA, Hugo HERBELIN et Amr SABRY. « A Type-Theoretic Foundation of Delimited Continuations ». Soumis, 2005.

[BB96] Franco BARBANERA et Stefano BERARDI. « A symmetric λ-calculus for classical program extraction ». *Information and Computation*, 125(2) :103–117, 1996.

[CFC58] Haskell B. CURRY, Robert FEYS et William CRAIG. *Combinatory Logic*, volume 1. North-Holland, 1958. §9E.

[CH94] Thierry COQUAND et Hugo HERBELIN. « Some remarks on Novikoff's calculus ». Non publié, 1994.

[CH00] Pierre-Louis CURIEN et Hugo HERBELIN. « The duality of computation ». Dans *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP 2000, Montreal, Canada, September 18-21, 2000*, SIGPLAN Notices 35(9), pages 233–243. ACM, 2000.

[Cro01] Tristan CROLARD. « Subtractive logic ». *Theor. Comput. Sci.*, 254(1-2) :151–185, 2001.

[dB78] Nicolas de BRUIJN. « A namefree lambda calculus with facilities for internal definition of expressions and segments ». Rapport Technique 78-WSK-03, Technological University Eindhoven, 1978.

[DF89] Olivier DANVY et Andrzej FILINSKI. « A Functional Abstraction of Typed Contexts ». Rapport Technique 89/12, DIKU, University of Copenhagen, Copenhagen, Denmark, août 1989.

[dG94] Philippe de GROOTE. On the Relation between the lambda-mu Calculus and the Syntactic Theory of Sequential Control. Dans F. PFENNING, éditeur, *Logic Programming and Automated Reasoning, Proc. of the 5th International Conference, LPAR'94*, pages 31–43. Springer, Berlin, Heidelberg, 1994.

[DG01] Mariangiola DEZANI-CIANCAGLINI et Elio GIOVANNETTI. « From Bohm's Theorem to Observational Equivalences : an Informal Account ». *Electr. Notes Theor. Comput. Sci.*, 50(2), 2001.

[DGL04] Daniel J. Dougherty, Silvia Ghilezan et Pierre Lescanne. « Characterizing strong normalization in a language with control operators ». Dans Eugenio Moggi et David Scott Warren, éditeurs, *Proceedings of the 6th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, 24-26 August 2004, Verona, Italy*, pages 155–166. ACM, 2004.

[DJS95] Vincent Danos, Jean-Baptiste Joinet et Harold Schellinx. LKQ and LKT : sequent calculi for second order logic based upon dual linear decompositions of the classical implication. Dans *Advances in Linear Logic*, volume 222, pages 211–224. Cambridge University Press, 1995.

[DJS97] Vincent Danos, Jean-Baptiste Joinet et Harold Schellinx. « A new deconstructive logic : Linear Logic ». *J. Symb. Log.*, 62(3) :755–807, 1997.

[DN05] René David et Karim Nour. « Arithmetical Proofs of Strong Normalization Results for the Symmetric lambda-mu-calculus ». Dans Pawel Urzyczyn, éditeur, *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, volume 3461 de *Lecture Notes in Computer Science*, pages 162–178. Springer, 2005.

[DP01] René David et Walter Py. « Lambda-mu-Calculus and Böhm's Theorem ». *J. Symb. Log.*, 66(1) :407–413, 2001.

[Fel88] Matthias Felleisen. « The Theory and Practice of First-Class Prompts ». Dans *Proceedings of the 15th ACM Symposium on Principles of Programming Languages (POPL '88)*, pages 180–190. ACM Press, New York, janvier 1988.

[FFKD86] Matthias Felleisen, Daniel P. Friedman, Eugene Kohlbecker et Bruce F. Duba. « Reasoning with continuations ». Dans *First Symposium on Logic and Computer Science*, pages 131–141, 1986.

[FH92] Matthias Felleisen et Robert Hieb. « The Revised Report on the Syntactic Theories of Sequential Control and State ». *Theor. Comput. Sci.*, 103(2) :235–271, 1992.

[Fil89a] Andrzej Filinski. « Declarative Continuations : an Investigation of Duality in Programming Language Semantics ». Dans David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts et Axel Poigné, éditeurs, *Category Theory and Computer Science*, volume 389 de *Lecture Notes in Computer Science*, pages 224–249. Springer, 1989.

[Fil89b] Andrzej Filinski. « *Declarative Continuations and Categorical Duality* ». Master thesis, DIKU, Danmark, août 1989.

[Fil94] Andrzej Filinski. « Representing Monads ». Dans *Conf. Record 21st ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'94, Portland, OR, USA, 17-21 Jan. 1994*, pages 446–457. ACM Press, New York, 1994.

[Fis72] Michael J. Fischer. « Lambda-calculus schemata ». Dans *Proc. ACM Conference on Proving Assertions About Programs*, volume 7(1) de *SIGPLAN Notices*, pages 104–109. ACM Press, New York, 1972.

[Fis93] Michael J. Fischer. « Lambda-Calculus Schemata ». *LISP and Symbolic Computation*, 6(3/4) :259–288, 1993. Reproduction de [Fis72].

[Gen35] Gerhard Gentzen. « Untersuchungen über das logische Schließen ». *Mathematische Zeitschrift*, 39 :176–210,405–431, 1935. Traduction anglaise dans [Sza69], « Investigations into logical deduction », pages 68-131.

[Gen38] Gerhard Gentzen. « Neue Fassung des Widerspruchsfreiheitsbeweises für die reine Zahlentheorie ». *Forschungen zur Logik und zur Grundlegung der exakter Wissenschaften, Neue Folge*, 4 :19–44, 1938. Traduction anglaise dans [Sza69], « New version of the consistency proof for elementary number theory », pages 252–286.

[Gir87] Jean-Yves Girard. « Linear Logic ». *Theor. Comput. Sci.*, 50 :1–102, 1987.

[Gir91] Jean-Yves Girard. « A New Constructive Logic : Classical Logic ». *Mathematical Structures in Computer Science*, 1(3) :255–296, 1991.

[GLT89] Jean-Yves Girard, Yves Lafont et Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[Gri90] Timothy G. Griffin. « The Formulae-as-Types Notion of Control ». Dans *Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90, San Francisco, CA, USA, 17-19 Jan 1990*, pages 47–57. ACM Press, New York, 1990.

[Her95] Hugo Herbelin. « A Lambda-Calculus Structure Isomorphic to Gentzen-Style Sequent Calculus Structure ». Dans Leszek Pacholski et Jerzy Tiuryn, éditeurs, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 de *Lecture Notes in Computer Science*, pages 61–75. Springer, 1995.

[Her97] Hugo Herbelin. « Games and Weak-Head Reduction for Classical PCF ». Dans Philippe de Groote et J. Roger Hindley, éditeurs, *Third International Conference on Typed Lambda Calculi and Applications, TLCA '97, Nancy, France, April 2-4, 1997, Proceedings*, volume 1210 de *Lecture Notes in Computer Science*, pages 214–230. Springer, 1997.

[Her01] Hugo Herbelin. « Explicit Substitutions and Reducibility ». *Journal of Logic and Computation*, 11(3) :431–451, 2001.

[Her05] Hugo Herbelin. « On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic ». Dans Pawel Urzyczyn, éditeur, *Seventh International Conference, TLCA '05, Nara, Japan. April 2005, Proceedings*, volume 3461 de *Lecture Notes in Computer Science*, pages 209–220. Springer, 2005.

[Hof95] Martin Hofmann. « Sound and Complete Axiomatisations of Call-by-Value Control Operators ». *Mathematical Structures in Computer Science*, 5(4) :461–482, 1995.

[How80] William A. Howard. The formulae-as-types notion of constructions. Dans *to H.B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980. Manuscrit non publié de 1969.

[HY97] Kohei Honda et Nobuko Yoshida. « Game Theoretic Analysis of Call-by-Value Computation ». Dans Pierpaolo Degano, Roberto Gorrieri et Alberto Marchetti-Spaccamela, éditeurs, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 de *Lecture Notes in Computer Science*, pages 225–236. Springer, 1997.

[KCR+98] Richard Kelsey, (ed.), William Clinger, (ed.), Jonathan Rees, (ed.), Hal Abelson, N. I. Adams IV, D. H. Bartley, G. Brooks, R. Kent Dybvig, Daniel P. Friedman, R. Halstead, Chris Hanson, Christopher T. Haynes, Eugene Kohlbecker, D. Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy L. Steele Jr., Gerald J. Sussman et Mitchell Wand. « Revised^5 Report on the Algorithmic Language Scheme ». *ACM SIGPLAN Notices*, 33(9) :26–76, 1998.

[KH03] Yukiyoshi Kameyama et Masahito Hasegawa. « A Sound and Complete Axiomatization of Delimited Continuations ». Dans Colin Runciman et Olin Shivers, éditeurs, *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25-29, 2003*, volume 38(9) de *SIGPLAN Notices*, pages 177–188. ACM Press, New York, 2003.

[Kri04] Jean-Louis Krivine. « Realizability in classical logic ». *Panoramas et synthèses*, 2004. À paraître.

[Lan65a] Peter Landin. « A correspondence between ALGOL 60 and Church's lambda-notation ». *Communications of the ACM*, 8 :89–101 and 158–165, 1965.

[Lan65b] Peter Landin. « A generalisation of jumps and labels ». Rapport Technique ECS-LFCS-88-66, UNIVAC Systems Programming Research, août 1965. Réimprimé dans `Higher Order and Symbolic Computation`, 11(2), 1998.

[Len03] Stéphane Lengrand. « Call-by-value, call-by-name, and strong normalization for the classical sequent calculus ». *Electr. Notes Theor. Comput. Sci.*, 86(4), 2003.

[Lik05] Silvia Likavec. « *Types for object oriented and functional languages* ». Phd thesis, Università degli Studi di Torino, février 2005.

[LRS93] Yves Lafont, Bernhard Reus et Thomas Streicher. « Continuations Semantics or Expressing Implication by Negation ». Rapport Technique 9321, Ludwig-Maximilians-Universität, München, 1993.

[Min96] Grigori MINTS. Normal forms for sequent derivations. Dans *Kreiseliana : About and Around George Kreisel*, pages 469–492. A.K. Peters, 1996.

[Mog88] Eugenio MOGGI. « Computational lambda-calculus and monads ». Rapport Technique ECS-LFCS-88-66, Edinburgh Univ., 1988.

[Mur92] Chetan R. MURTHY. « A Computational Analysis of Girard's Translation and LC ». Dans *Proc. of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 90–101, Santa Cruz, CA, 1992.

[Nip91] Tobias NIPKOW. « Higher-Order Critical Pairs ». Dans *Proc. 6th IEEE Symp. Logic in Computer Science*, pages 342–349. IEEE Press, 1991.

[Ong96] C.-H. Luke ONG. « A semantic view of classical proofs : type-theoretic, categorical, denotational characterizations ». Dans *Proceedings of 11th IEEE Annual Symposium on Logic in Computer Science*, pages 230–241. IEEE Computer Society Press, 1996.

[OS97] C.-H. Luke ONG et Charles A. STEWART. « A Curry-Howard Foundation for functional computation with control ». Dans *Proceedings of ACM SIGPLAN-SIGACT Symposium on Principle of Programming Languages, Paris, January 1997*, pages 215–227. ACM Press, 1997.

[Par92] Michel PARIGOT. « Lambda-mu-calculus : An algorithmic interpretation of classical natural deduction ». Dans *Logic Programming and Automated Reasoning : International Conference LPAR '92 Proceedings, St. Petersburg, Russia*, pages 190–201. Springer-Verlag, 1992.

[Par00] Michel PARIGOT. « Strong Normalization of Second Order Symmetric lambda-Calculus ». Dans Sanjiv KAPOOR et Sanjiva PRASAD, éditeurs, *Foundations of Software Technology and Theoretical Computer Science, 20th Conference, FST TCS 2000 New Delhi, India, December 13-15, 2000, Proceedings*, volume 1974 de *Lecture Notes in Computer Science*, pages 442–453. Springer, 2000.

[Pie91] Benjamin C. PIERCE. « Programming with intersection types, union types, and polymorphism ». Rapport Technique CMU-CS-91-106, Carnegie Mellon University, février 1991.

[Plo75] Gordon D. PLOTKIN. « Call-by-name, call-by-value and the lambda-calculus ». *Theor. Comput. Sci.*, 1 :125–159, 1975.

[Pol03] Emmanuel POLONOVSKI. « *Subsitutions explicites, logique et normalisation* ». Thèse de doctorat, Université Paris 7, juin 2003.

[PR01] David PYM et Eike RITTER. « On the Semantics of Classical Disjunction ». *Journal of Pure and Applied Algebra*, 159 :315–338, 2001.

[Pra65] Dag PRAWITZ. *Natural Deduction, a Proof-Theoretical Study.* Almqvist and Wiksell, Stockholm, 1965.

[Rau74] Cecylia RAUSZER. « Semi-boolean algebras and their application to intuitionistic logic with dual connectives ». *Fundamenta Mathematicae*, 83 :219–249, 1974.

[Rey72] John C. REYNOLDS. « Definitional interpreters for higher-order programming languages ». Dans *ACM '72 : Proceedings of the ACM annual conference*, pages 717–740, New York, NY, USA, 1972. ACM Press.

[Roc05] Jérôme ROCHETEAU. « $\lambda\mu$-calculus and duality : call-by-name and call-by-value ». Dans Jürgen GIESL, éditeur, *Term Rewriting and Applications*, volume 3467 de *Lectures Notes in Computer Science*, pages 204–218. Springer-Verlag, mars 2005.

[Sel01] Peter SELINGER. « Control categories and duality : on the categorical semantics of the lambda-mu calculus ». *Mathematical Structures in Computer Science*, 11(2) :207–260, 2001.

[Sel03] Peter SELINGER. « Some remarks on Control categories ». Accessible sur la page web de l'auteur, 2003.

[SF90] Dorai SITARAM et Matthias FELLEISEN. « Reasoning with continuations II : full abstraction for models of control ». Dans *LFP '90 : Proceedings of the 1990 ACM conference on LISP and functional programming*, pages 161–175. ACM Press, 1990.

[SF93] Amr SABRY et Matthias FELLEISEN. « Reasoning about Programs in Continuation-Passing Style ». *Lisp and Symbolic Computation*, 6(3-4) :289–360, 1993.

[SG93] Guy L. STEELE JR. et Richard P. GABRIEL. « The evolution of Lisp ». Dans *HOPL-II : The second ACM SIGPLAN conference on History of programming languages*, pages 231–270, New York, NY, USA, 1993. ACM Press.

[Sha04] Chung-chieh SHAN. « Shift to Control ». Dans Olin SHIVERS et Oscar WADDELL, éditeurs, *Proceedings of the 5th workshop on Scheme and Functional Programming*, pages 99–107, 2004.

[SS75] Gerald J. SUSSMAN et Guy L. STEELE JR.. « Scheme : An Interpreter for Extended Lambda Calculus ». Rapport Technique AIM-349, Massachusetts Institute of Technology, Cambridge, MA, USA, 1975. Réimprimé dans `Higher Order and Symbolic Computation`, 11(4), 1998.

[Sza69] Manfred E. SZABO, éditeur. *The Collected Works of Gerhard Gentzen*. North Holland, Amsterdam, 1969.

[Thi98] Hayo THIELECKE. « An Introduction to Landin's "A Generalization of Jumps and Labels" ». *Higher Order and Symbolic Computation*, 11(2) :117–123, 1998.

[UB01] Christian URBAN et Gavin M. BIERMAN. « Strong Normalisation of Cut-Elimination in Classical Logic ». *Fundamenta Informaticae*, 45(1-2) :123–155, janvier 2001.

[Wad03] Philip WADLER. « Call-by-value is dual to call-by-name ». Dans Colin RUNCIMAN et Olin SHIVERS, éditeurs, *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25-29, 2003*, volume 38(9) de *SIGPLAN Notices*, pages 189–201. ACM, 2003.

Index