


Université Paris XIII — Sorbonne Paris Nord
École doctorale Sciences, Technologies, Santé Galilée

Vers une théorie générale des Approximations dans les Langages de Programmation

Thèse de Doctorat
présentée par

Aloÿs Dufour
Laboratoire d'Informatique de Paris-Nord (LIPN)

pour l'obtention du grade de
Docteur en Informatique

soutenue le 15 décembre 2025 devant le jury d'examen constitué de :

Mimram	Samuel	École Polytechnique	Examinateur
Guerrieri	Giulio	Université de Sussex	Rapporteur
Manzonetto	Giulio	Université Paris-Cité	Rapporteur
Faggian	Claudia	CNRS, Université Paris-Cité	Examinatrice
Kerjean	Marie	CNRS, USPN	Examinatrice
Mazza	Damiano	CNRS, USPN	Directeur de thèse

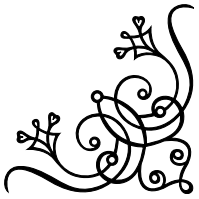



Table des matières

Introduction

Motivations

Le λ-calcul est une pierre angulaire de l'informatique théorique, plus précisément de la calculabilité, de la théorie des langages de programmation dits fonctionnels et de la logique. Sa structure intrinsèque repose seulement sur la notion de fonction et de substitution. Cette simplicité permet de le décliner en de nombreuses variantes se prêtant à l'analyse ou la modélisation d'autant de systèmes. Il a permis une formalisation des systèmes de types, et c'est par ce biais qu'est arrivée la correspondance de Curry-Howard entre les preuves en logique mathématique, et les programmes en informatique.

État de l'art

Durant ces dernières décennies, l'idée qu'un langage de programmation puisse être approximé par le biais de du λ-calcul linéaire et multilinéaire a vu son nombre d'applications augmenter. L'origine, sans doute l'exemple le plus connu, de cette idée est le développement de Taylor du λ-calcul non-typé via le λ-calcul à ressources, introduit par [27, 29]. Ce fut le point de départ d'une longue série de résultats comme en sémantique [13], λ-calcul pur [5] ainsi qu'en calculabilité probabiliste [30, 49].

Une autre source importante d'applications provient du lien entre approximations et théorie des types intersection, pour la première fois décrite par de

Carvalho pour le cas non-idempotent [14], et étendu par la suite par Mazza *et al.* pour couvrir le cas idempotent [42]. De cette manière, l'idée de de Carvalho d'utiliser les types intersection pour inférer des bornes exactes sur le temps d'exécution de programme [13] peut être étendu dans des contextes plus larges, permettant l'étude de l'utilisation mémoire [2, 4, 3], l'analyse de programmes concurrents [17], ou encore allant jusqu'à prouver le théorème de Cook-Levin en utilisant des systèmes de types [39].

Sujet de la thèse

Dans [38] il est proposé une approche axiomatique aux systèmes d'approximations de langages de programmation. L'idée est la suivante : une relation d'approximation $t \sqsubset M$ entre un programme approximant t et un programme M devrait induire une « adjonction » entre calculs et approximations de calculs, pour tout programme M et approximation u,

$$\frac{M \text{ s'évalue en } N \text{ tel que } u \sqsubset N}{\text{il existe } t \sqsubset M \text{ tel que } t \text{ s'évalue en } u} \text{ ssi}$$

ou, diagrammatiquement,

$$\begin{array}{ccc} & u & \\ & \sqcap & \\ M & \longrightarrow & N \end{array} \quad \Leftrightarrow \quad \begin{array}{ccc} t & \longrightarrow & u \\ \sqcap & & \\ M & & \end{array}$$

où les flèchent représentent l'évaluation des programmes. L'intuition voudrait que si l'on considère les approximations comme des morceaux d'information, et que l'on lit $t \sqsubset M$ comme « M contient l'information t », une relation d'approximation assure qu'un programme M s'évalue en quelque chose contenant le morceau d'information u si et seulement si une approximation de M s'évalue en u lui-même. On pourra remarquer une analogie avec la continuité au sens topologique.

Le formalisme axiomatique encapsule toutes les instances d'approximation de programmes connues jusqu'à maintenant, et permet d'en considérer

bien d'autres. Il subsiste cependant une question importante bien qu'informelle : *d'où viennent les approximations ?*

Le point de départ de cette thèse est l'idée que, dans le contexte des langages de programmation, les approximations ont une origine géométrique. Plus précisément, une relation d'approximation $t \sqsubset M$ devrait provenir de l'existence d'une sorte de *morphisme étale* de t à M. La notion de morphisme étale est bien connue dans différents contexte géométriques (variétés différentielles, schémas, topoi...) comme reformulation adaptée de la notion d'homéomorphisme local de la topologie.

Les intuitions ci-dessus résultent des observations que l'ont fait en regardant les approximations de programmes : dans toutes les notions d'approximations, $t \sqsubset M$ est vrai précisément lorsque la construction syntaxique t s'envoie sur une construction syntaxique similaire de M, respectant la structure syntaxique proche et, de plus, pour chaque construction syntaxique de M, il peut y avoir plusieurs constructions correspondantes à t. Ce qui semble indiqué que t est un « espace étalé » au-dessus de M.

Il est à noter que, pour que le point de vue ci-dessus ait du sens, les programmes approximant et approximés doivent vivre dans le même monde, dans le cas contraire on ne pourrait pas parler de morphismes entre eux. Cela peut d'ailleurs être surprenant au premier abord : prenons l'exemple primordial d'Ehrhard et Regnier des développements de Taylor [27, 29], les approximations sont des λ-termes à ressources et les programmes approximés sont des λ-termes, et vivent donc dans deux mondes calculatoires avec des sémantiques opérationnelles bien distinctes. Cependant, comme montré dans le λ-calcul affine infinitaire [41], il est possible de voir un programme usuel M comme un programme infinitaire de la même nature qu'un programme approximant t, de sorte qu'il n'y a pas d'inconsistance dans le fait de prendre approximant et approximés vivant dans le même monde ; utiliser différent langages de programmation relève davantage de commodité que de nécessité conceptuelle.

Contributions

Premièrement, nous avons introduit un calcul, $\mathcal{P}roc$, avec une syntaxe suffisamment riche pour exprimer des relations d'approximations en son sein.

Deuxièmement, nous avons développé les notions d'approximations de Böhm et de Taylor pour $\mathcal{P}roc$ jusqu'à atteindre le théorème de commutation entre les deux, pour la première fois énoncé par Ehrhard et Regnier [27].

Troisièmement, suivant l'intuition géométrique, nous avons introduit puis utilisé une notion de plongement afin de « tirer en arrière » à la fois ces notions d'approximations, mais également ce théorème central.

Quatrièmement, nous donnons des plongements de différents calculs dans $\mathcal{P}roc$:

- le $\lambda!$-calcul, pour *call-by-push-value*,
- le calcul des piles pour la logique classique,
- le π-calcul polyadique asynchrone,

afin d'appliquer notre formalisme et de déduire quelles sont les différentes notions d'approximations associées, et de constater qu'elles coïncident avec les notions déjà connues [27, 29, 44].

Plan de la thèse

Dans cette thèse sont présentés des résultats sur une version du théorème de commutation Böhm-Taylor appliqués à un système calculatoire forgé pour l'occasion, $\mathcal{P}roc$, plus général, ainsi que la définition de plongements entre des calculs connus dans $\mathcal{P}roc$ permettant de retrouver ce théorème dans des cadres connus.

Cette thèse est organisée de la manière suivante :

- Le chapitre 1 : introduit quelques définitions, rappels et notations sur les concepts logiques de base utilisés dans cette thèse. À savoir le λ-calcul ainsi que la logique linéaire qui sont les points de départs.
- Le chapitre 2 : présente les deux systèmes d'approximations en λ-calcul, de Böhm et de Taylor, et esquisse une présentation axiomatique de ce

qui constitue un système d'approximation en λ-calcul.

- Le chapitre 3 : introduit la notion de plongement d'un calcul dans un autre, et présente le calcul de processus $\mathcal{P}roc$, inspiré du π-calcul et de la logique linéaire différentielle. Nous utilisons cette notion de plongement pour établir les résultats généraux sur le théorème de commutation.
- Le chapitre 4 : utilise les résultats du chapitre précédent pour les appliquer à différents calculs — logique linéaire et types intersections, *call-by-push-value*, logique classique et une certaine variante de calcul de processus — pour retrouver les versions du théorème de commutation connus dans ces calculs.

Liste de publications

Les résultats des chapitres 3 et 4 ont fait l'objet d'une publication :

- Aloÿs Dufour et Damiano Mazza. « Böhm and Taylor for All ! » In : *Proceedings of FSCD*. 2024, 26 :1-26 :20

Liste de publications

Chapitre 1
Préliminaires

Nous commençons ici par donner quelques définitions & résultats classiques en lambda-calcul —introduit dans les années 1930 par Church— ainsi qu'en logique linéaire —introduite en 1987 par Girard [32]— qui sont à la base de nos travaux, et plus généralement de notre domaine d'étude.

1.1 Lambda-calcul

La problématique de savoir quelle fonction est calculable fut centrale dans les mathématiques du début du XIXe siècle. Elle s'est vue obtenir au moins trois réponses largement connues et étudiées encore aujourd'hui : le λ-calcul de Church, les fonctions μ-récursives de Gödel, et les machines de Turing. Ces trois formalismes sont les trois piliers fondateurs de l'informatique théorique et sont équivalents. Cela a amené la formulation de *l'hypothèse de Church* de la part de Kleene : tout formalisme permettant une calculabilité effective, physique, est au plus aussi expressive que l'un des trois piliers précédents. Cette hypothèse vérifiée jusqu'à présent, même avec des systèmes calculatoires quantiques plus subtiles que l'électronique classique [35].

Le principe du λ-calcul est le suivant : tout « λ-terme » peut-être vu comme une fonction, et formalise l'habitude mathématico-calculatoire de faire du « chercher-remplacer » —pour utiliser un vocabulaire d'informatique pratique— de variables ou de termes dans d'autres termes.

Soit $\mathcal{V}$ un ensemble infini dénombrable utilisé pour les variables.

Définition 1.1. — *L'ensemble* Λ *des* λ-termes *sur* $\mathcal{V}$*, est défini par la grammaire grammaire 1.1.*

$$\begin{array}{rcll} M & ::= & v & \textit{variable} \\ & | & M(N) & \textit{application} \\ & | & \lambda v.M & \textit{abstraction} \end{array}$$

Grammaire 1.1 : λ-calcul pur.

Convention 1.2. — *Afin d'alléger la notation de nos* λ*-termes, introduisons quelques conventions habituelles :*

1. *les applications s'associent à gauche, et sont prioritaires sur les abstractions, par exemple* MNP *signifie* $(MN)P$ *;*
2. *les abstractions peuvent s'écrire en séquences, par exemple* $\lambda xy.M$ *signifie* $\lambda x.\lambda y.M$*, ou encore si* $\vec{x} = (x_i)_{1 \leqslant i \leqslant n}$*,* $\lambda \vec{x}.M$ *signifie* $\lambda x_1 \dots \lambda x_n.M$*.*

Notations 1.3. — *Soit* $M \in \Lambda$*,*

- $\mathsf{fn}(()M)$ *désigne les* variables libres*, ou* nom libres *du* λ*-terme* M*, définit selon les cas*
  - *si* $x \in V$*,* $\mathsf{fn}(()x) = \{x\}$ *;*
  - *si* $M, N \in \Lambda$*,* $\mathsf{fn}(()MN) = \mathsf{fn}(()M) \cup \mathsf{fn}(()N)$ *;*
  - *si* $x \in V$ *et* $M \in \Lambda$*,* $\mathsf{fn}(()\lambda x.M) = \mathsf{fn}(()M) - \{x\}$*.*
- *L'ensemble des* λ*-termes* clos *(sans variable libre), dits* combinateurs*, est* $\Lambda^{\varnothing} := \{M \in \Lambda, \mathsf{fn}(()M) = \varnothing\}$*.*
- *Les variables non-libres sont dites* liées*, et sont susceptibles d'être* renommées*. Puisque* y *est lié dans* $\lambda y.M \in \Lambda$*, on peut remplacer* y *en* z *:* $\lambda z.M\{y/z\}$*, où* $M\{y/z\}$ *désigne la* substitution *de toutes les occurrences de* y *dans* M *par* z*. Le renommage de variables liées constitue une relation d'équivalence sur les* λ*-termes, l'*α-équivalence*, ou encore* α-conversion*, par laquelle on identifiera les* λ*-termes, et* Λ *désignera par la suite l'ensemble des* λ*-termes à* α*-équivalence près.*

Définition 1.4. — *La* β-réduction, *notée* $\to_\beta$, *est définie comment la clôture contextuelle de la règle*

$$(\lambda x.M)N \to M\{N/x\}$$

où $M\{N/x\}$ *désigne la substitution simultanée dans* M *de toutes les occurrences libres de* x *par* N *(usant de l'*α*-conversion dans* M *pour éviter la capture de variables libres de* N*).*

Notations 1.5 (De réécriture). — *Pour toute relation, notamment* $\to_\beta \subset \Lambda \times \Lambda$, *nous introduisons*

- $\to_\beta^n$ *désigne une suite de* $n \in \mathbb{N}$ *relations ;*
- $\to_\beta^*$ *la* clôture transitive *de* $\to_\beta$ *(suite finie d'un nombre arbitraire de réductions);*
- $=_\beta$ *la* β-conversion, *qui est la clôture symétrique de* $\to_\beta^*$.

Remarque 1.6. — *La* β*-réduction induit une certaine dynamique dans le* λ*-calcul, correspondant en apparence à de simples suites de substitutions, mais suffisant à l'inclure dans les modèles de calcul les plus expressifs. Les propriétés de cette dynamique sont bien connues :*

- *confluence (Church-Rosser)*
- *existence d'une* forme normale *pour les réductions qui terminent (pour* $M \in \Lambda$ *notée* $\mathrm{nf}_\beta(M)$*);*
- *la forme normale* de tête *de* $M \in \Lambda$ *est notée* $\mathrm{hnf}_\beta(M)$.

Exemples 1.7. — *Les* λ*-termes représentants à la fois des programmes, des fonctions et des calculs à effectuer, sans compter le développement historique et la culture du domaine de recherche engendré, certains se voient attribuer des notations standards.*

- $I = \lambda x.x$, *l'identité, le neutre pour la composition ;*
- $K = \lambda xy.y$, *la projection sur la deuxième variable, combinateur historique SKR, encode le booléen* faux *;*
- $T = \lambda xy.x$, *projection sur la première variable, encode le booléen* vrai *;*

- $\Delta = \lambda x.xx$, *opérateur qui prend une entrée pour l'appliquer à elle-même, où l'on se rend combinatoirement compte que l'évolution de la taille des termes est une question non-triviale ;*
- $\Omega = \Delta\Delta$, *le terme le plus simple se réduisant à lui-même ;*
- $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, *combinateur de point fixe de* Turing *;*
- $\Theta = (\lambda fx.x(ffx))(\lambda fx.x(ffx))$, *l'autre combinateur de point fixe.*

1.2 Logique linéaire

Introduite par [32] et développée par la suite principalement par l'école française, italienne, et anglaise de logique, la *logique linéaire* est une raffinement de la logique intuitionniste permettant notamment un suivi fin du nombre d'utilisations des formules.

On pourra se référer à [32] et [33] pour des références historiques, [43] pour la présentation catégorique, et [36].

Définition 1.8 (Formules de LL). — *Les formules de la logiques linéaires sont construites avec les atomes (variables) et les unités (des connecteurs), à l'aide des connecteurs, modalités et quantificateurs :*

- *un* atome *est une variable propositionnelle (i.e. du second ordre)* α *ou son dual* $\alpha^\perp$*, plus généralement, c'est un prédicat atomique* $\alpha(t_1, \dots, t_n)$ *ou son dual* $\alpha(t_1, \dots, t_n)$ *où les* t_i *sont des termes du premier ordre ;*
- *les* connecteurs multiplicatifs *sont* $\otimes$ *(*tenseur*, ou* conjonction multiplicative*) et son dual* $\parr$ *(*par*, ou* disjonction multiplicative*); les unités correspondantes sont* $\mathbf{1}$ *et* $\bot$ *;*
- *les* connecteurs additifs *sont* $\&$ *(*avec*, ou* conjonction additive*), et son dual* $\oplus$ *(*plus*, ou* disjonction additive*); les unités correspondantes sont* $\top$ *et* $\mathbf{0}$ *;*
- *les* modalités exponentielles *sont* ! *(*bien-sûr*) et son dual* ? *(*pourquoi-pas*);*

- *les* quantificateurs *sont* $\forall$ *et son dual* $\exists$*, s'appliquant à des variables du premier ou second ordre*

Si A *est un atome,* $A^\perp$ *est son dual, en particulier* $A^{\perp\perp} = A$*. La* négation linéaire *est étendue à toutes les formules par les lois de* de Morgan*.*
*L'*implication linéaire *est définie par* $A \multimap B := A^\perp \parr B$*.*

Définition 1.9 (Séquents de LL). — *Les* séquents *sont de la forme* $\vdash \Gamma$ *où* Γ *est une suite de formule* $(A_i)_{1\leqslant i\leqslant n}$*, avec* $n \in \mathbb{N}$*. En pratique, on identifie* $\vdash \Gamma$ *et* Γ*, et l'on écrit* $\Gamma^\perp$ *pour* $(A_i^\perp)_{1\leqslant i\leqslant n}$*, de même pour les modalités exponentielles, de plus, si* Δ *est une autre suite de formules,* $\Gamma \vdash \Delta$ *est synonyme à* $\vdash \Gamma^\perp, \Delta$*.*
Un séquent est prouvable *lorsqu'il peut être dérivé en utilisant les règles suivantes :*

$$\frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}\ ex \qquad \frac{}{\vdash A, A^\perp}\ id \qquad \frac{\vdash A, \Gamma \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ cut$$

$$\frac{\vdash A, \Gamma \quad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta}\ \otimes \qquad \frac{\vdash A, B, \Gamma}{\vdash A \parr B, \Gamma}\ \parr \qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1} \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ \bot$$

$$\frac{\vdash A, \Gamma \quad \vdash B, \Gamma}{\vdash A \,\&\, B, \Gamma}\ \& \qquad \frac{\vdash A, \Gamma}{\vdash A \oplus B, \Gamma}\ \oplus_1 \qquad \frac{\vdash B, \Gamma}{\vdash A \oplus B, \Gamma}\ \oplus_2 \qquad \frac{}{\vdash \top, \Gamma}\ \top$$

$$\frac{\vdash A, ?\Gamma}{\vdash !A, ?\Gamma}\ !p \qquad \frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}\ ?d \qquad \frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}\ ?c \qquad \frac{\vdash \Gamma}{\vdash ?A, \Gamma}\ ?w$$

$$\frac{\vdash A, \Gamma}{\vdash \forall \xi.A, \Gamma}\ \forall \qquad \frac{\vdash A[\tau/\xi], \Gamma}{\vdash \exists \xi.A, \Gamma}\ \exists$$

Remarques 1.10. —

- *La règle d'échange est la seule règle structurelle.*
- *Les règles des modalités exponentielles s'appellent respectivement, la* promotion*, la* déréliction*, la* contraction *et l'*affaiblissement*.*
- *Dans la règle* $\forall$*,* ξ *ne doit pas avoir d'occurrence libre dans* Γ*.*
- *Dans la règle* $\exists$*,* τ *est un terme du premier ordre (si* ξ *est une variable du premier ordre) ou une formule (si* ξ *est une variable du second ordre).*

- *Par la règle d'échange, toute permutation de* Γ *peut être dérivée de* Γ*, il est donc courant qu'en pratique les séquent sont considérés comme des multi-ensembles finis, et la règle d'échange est implicite.*

Notations I.II (Terminologie). —

- *LL : Le fragment propositionnel complet de la Logique Linéaire.*
- *MLL : Le fragment multiplicatif de la logique linéaire (sans exponentiels).*
- *MALL : Les fragments additifs et multiplicatifs de la logique linéaire (sans exponentiels).*
- *MELL : Le fragments multiplicatif de la logique linéaire avec les exponentiels.*

Chapitre 2
Sur les systèmes d'approximations

Que cela soit en analyse ou en langages de programmation, les approximations permettent un autre point de vue sur les objets et l'établissement de propriétés. On pensera aux approximations de fonctions continues par des polynômes via la formule de Taylor que l'on croise dès nos jeunes années d'études. Il en va de même en sémantique, aussi bien qualitative que quantitative, comme nos deux exemples en λ-calcul le montrent : les approximations de Böhm et de Taylor. La troisième section de se chapitre introduit une notion axiomatique de ce qu'est une approximation, ou plutôt un système d'approximations, dans un langage de programmation.

2.1 *Exemples*

2.1.1 Approximations de Böhm

Les arbres de Böhm [7, 9] et leurs variantes sont un élément essentiel de la sémantique du λ-calcul. Intuitivement, il s'agit de formes normales éventuellement infinies représentant l'essence du comportement d'un λ-terme. En tant que tel, ils peuvent être considérés comme les limites d'un ensemble d'approximations finies, appelées *approximations de Böhm*, décrivant des parties de plus en plus grandes de l'arbre de Böhm.

Il est intéressant de noter que les arbres de Böhm habituels, ainsi que les

arbres de Böhm appel-par-valeur, peuvent également être considérés comme résultant d'une notion générale d'« arbre de Böhm » pour la logique linéaire, ramenée le long des codages appel-par-nom et appel-par-valeur de Girard. Il est naturel de se demander si les différentes formes du théorème de commutation sont également des « tirés-en-arrière » d'un théorème de commutation plus général.

2.1.2 Approximations de Taylor

Plus récemment, Ehrhard et Regnier ont introduit une autre notion d'approximation pour le λ-calcul, qui sous-tend leur *expansion de Taylor* [27, 29]. Basée sur l'idée que les programmes peuvent être considérés comme des fonctions analytiques sur certains espaces vectoriels topologiques [24, 23], cette notion peut être allégée en oubliant ses aspects quantitatifs (les coefficients de la série de Taylor) et, comme un arbre de Böhm, être présentée comme un ensemble d'approximations finies. Ces *approximations de Taylor* sont syntaxiquement très différentes des approximations de Böhm : ce ne sont pas nécessairement des formes normales, et elles sont linéaires, au sens de la logique linéaire [32] (elles sont apparentées aux *termes avec multiplicités* de Boudol [10]), de sorte que leur durée d'exécution est limitée par leur taille.

Étant donné un λ-terme t, puisqu'une approximation de Böhm est, essentiellement, aussi un λ-terme, il est possible de prendre l'ensemble de toutes les approximations de Taylor de toutes les approximations de Böhm de t, ce qui donne un ensemble $\mathcal{T}(\mathrm{BT}(t))$. D'autre part, comme les approximations de Taylor se normalisent toujours, il est possible de prendre l'ensemble de toutes les formes normales de toutes les approximations de Taylor de t, ce qui donne un ensemble $\mathrm{NF}(\mathcal{T}(t))$. Un résultat clé d'Ehrhard et Regnier, connu sous le nom de *théorème de commutation*, stipule que $\mathcal{T}(\mathrm{BT}(t)) = \mathrm{NF}(\mathcal{T}(t))$. Ce lien entre les approximations de Böhm et de Taylor est un outil étonnamment puissant, qui implique une multitude de théorèmes fondamentaux dans le λ-calcul pur [6].

La théorie des approximations de Taylor a été étendue à l'appel-par-valeur [22], au *call-by-push-value* [15] et au $\lambda\mu$-calcul [5]. Dans le cadre de l'appel par valeur, où une notion d'arbre de Böhm est disponible, on sait que le théorème

de commutation s'applique [44]. Ces résultats exploitent le fait qu'en réalité, la notion d'expansion de Taylor existe dans le cadre beaucoup plus général de la *logique linéaire différentielle* [20]. De manière informelle, chaque fois qu'un système $\mathcal{S}$ peut être codé en logique linéaire différentielle, une notion d'approximation de Taylor pour $\mathcal{S}$ peut être "tirée-en-arrière" le long du codage.

2.2 Généralisation

2.2.1 Formalisme bicatégorique

Définition 2.1. — *Une* double catégorie posetale *est un objet en catégorie dans* Pos*, la catégorie des ensembles partiellement ordonnés avec applications croissantes.*

Remarque 2.2. — *La donnée d'une telle catégorie,* $\mathfrak{D}$ *consiste en :*

(i) un ensemble ordonné $(\mathfrak{D}_0, \leqslant_0)$ *dont les éléments sont appelés les objets,*
(ii) un ensemble ordonné $(\mathfrak{D}_1, \leqslant_1)$ *de flèches entre les objets, dont la composition dénotée par* $\cdot$*, est strictement associative et a des éléments neutres strictes* id_a*, tels que*

- *si* $f : a \to a'$*,* $g : b \to b'$ *et* $f \leqslant_1 g$*, alors* $a \leqslant_0 b$ *et* $a' \leqslant_0 b'$*,*
- *si* $a \leqslant_0 b$*, alors* $\mathrm{id}_a \leqslant_1 \mathrm{id}_b$*,*
- *si* $f : a \to b$*,* $g : b \to c$*,* $f' : a' \to b'$*,* $g' : b' \to c'$ *et* $f \leqslant_1 f'$*,* $g \leqslant_1 g'$*, alors* $g \cdot f \leqslant_1 g' \cdot f'$*.*

Une telle catégorie a deux opposées : une opposée verticale*,* $\mathfrak{D}^{\mathrm{co}}$*, dans laquelle l'ordre des ensembles ordonnés* $\mathfrak{D}_0$ *et* $\mathfrak{D}_1$ *est inversé, ainsi qu'une opposée* horizontale*,* $\mathfrak{D}^{\mathrm{op}}$*, dans laquelle la direction des flèches est inversée.*

Définition 2.3. — *Un foncteur entre doubles catégories posetales* $F : \mathfrak{D} \to \mathfrak{D}'$ *est une paire* $(F_0 : \mathfrak{D}_0 \to \mathfrak{D}'_0, F_1 : \mathfrak{D}_1 \to \mathfrak{D}'_1)$ *d'applications croissantes satisfaisants les conditions de fonctorialités habituelles et attendues :*

(i) si $f : a \to b$*, alors* $F_1(f) : F_0(a) \to F_0(b)$*,*
(ii) $F_1(g \cdot f) = F_1(g) \cdot (f)$ *et* $F_1(\mathrm{id}_a) = \mathrm{id}_{F_0(a)}$*.*

Remarque 2.4. — *Notons DbPos la catégorie des doubles catégories posetales muni de ses foncteurs, elle hérite des produits de Pos.*

Définition 2.5. — *Une transformation naturelle θ entre deux tels foncteurs $F, G : \mathfrak{C} \to \mathfrak{D}$ de doubles catégories posetales est une famille de flèches $(\theta_a)_{a \in \mathfrak{C}_0}$ avec $\theta_a a : Fa \to Ga$, $\forall a, a' \in \mathfrak{C}_0$, $a \leqslant_0 a' \Rightarrow \theta_a \leqslant_1 \theta_{a'}$, et qui est de plus naturel au sens usuel du terme.*

Remarque 2.6. — *Soient deux foncteurs $F, G : \mathfrak{C} \to \mathfrak{D}$, on notera $G \leqslant F$ lorsque $Fa \leqslant_0 Ga$ pour tout $a \in \mathfrak{C}_0$ et $Ff \leqslant Gf$ pour tout $f \in \mathfrak{C}_1$.*

Une double catégorie posetale $\mathfrak{D}$ est dite complètement dirigé *lorsque $(\mathfrak{D}_0, \leqslant_0)$ et $(\mathfrak{D}_1, \leqslant_1)$ sont des DCPOs.*

Définition 2.7. — *Soit $(D, \leqslant)$ un ordre partiel,*

- *un sous-ensemble $\Delta \subseteq D$ est* dirigé *lorsqu'il est non-vide et $\forall x, y \in \Delta$, $\exists z \in \Delta$, $x \leqslant z$ et $y \leqslant z$,*
- *$(D, \leqslant)$ est* complètement dirigé *(DCPO) lorsque tous les sous-ensembles dirigés de D ont un supremum dans D, noté $\bigvee \Delta$,*
- *si $(D, \leqslant)$ est un DCPO et qu'il a un élément minimum $\bot \in D$, alors c'est un* ordre partiel complet *(CPO).*

Remarque 2.8. — *Toute double catégorie posetale ne peut être complétée, mais un critère d'existence de la complétion est donné.*

Lemme 2.9. — *Une double catégorie posetale est complétable si*

$$\forall b \leqslant_0 a, \quad \forall p : b \to b', \quad \exists \widehat{p} : a \to a', \quad p \leqslant_1 \widehat{p},$$
$$\forall r : c \to c', \quad p \leqslant_1 r, \quad \exists q : a' \to a'_1, \quad \mathrm{id}_{b'} \leqslant_1 q, \quad q \cdot \widehat{p} \leqslant_1 r,$$

(et donc $p \leqslant_1 \widehat{p} \cdot q \leqslant_1 r$).

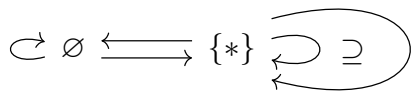
Remarque 2.10. — *Ce qui peut se traduire par « les flèches horizontales sont monotones », et, de plus, chaque flèche peut-être « sur-approximée » d'une manière minimale.*

La propriété devient particulièrement flagrante si l'on interprète les objets comme des programmes et les flèches comme des computations : si un programme b effectue un calcul p, alors une sur-approximation a de b effectue une sur-approximation $\widehat{p}$ de p (car le calcul est monotone), qui est minimale dans le sens

où n'importe quelle sur-approximation de p partant d'une sur-approximation b « plus grosse » que a est en fait une sur-approximation d'une extension de $\widehat{p}$.

Une double catégorie posetale complétable $\mathfrak{D}$ admet une completion idéale *$\widehat{\mathfrak{D}}$, qui est complètement dirigée, et il y a un plongement pleinement fidèle $y : \mathfrak{D} \to \widehat{\mathfrak{D}}$.*

Exemple 2.11. — *Dans Rel,*



Définition 2.12. — *Un* système d'approximations *est un foncteur entre doubles catégories posetales*

$$\mathrm{Apx} : \mathcal{A}^{\mathrm{co}} \times \mathcal{L} \to \mathfrak{D}$$

où $\mathfrak{D}$ est la sous-catégorie de Rel introduite tantôt, $\mathcal{L}$ une petite catégorie, $\mathcal{A}$ petite et complétable, et telles que la condition suivante soit satisfaite. Soit un foncteur $T : \mathcal{L}^{\mathrm{op}} \to$ Rel tel que

- *sur les objets,*

$$T(M) := \{t \in \mathfrak{A}_0 \mid \mathrm{Apx}(t, M) = 1\},$$

- *sur les flèches,*

$$T(M \xrightarrow{f} N) := \{(u, t) \in T(N) \times T(M) \mid \exists t \xrightarrow{a} u.\mathrm{Apx}(a, f) = \mathrm{id}_1\}.$$

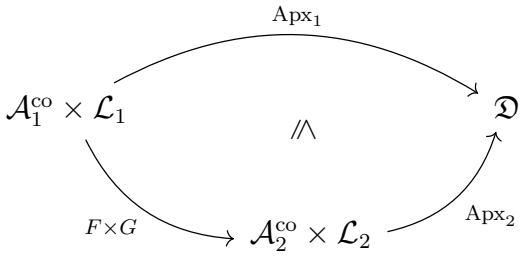
Remarque 2.13. — *On peut vérifier que T est en effet un foncteur. Pour que* Apx *soit un système d'approximations, il est nécessaire que :*

(i) $\forall M \in \mathcal{L}_0$*, l'ensemble $T(M)$ est dirigé (en tant que sous-ensemble ordonné de $\mathcal{A}_0$),*

(ii) $\forall f : M \to N$*, la relation $T(f) \subseteq T(N) \times T(M)$ est entière, c'est-à-dire que $\forall u \in T(N)$, $\exists t \in T(M)$ tel que $(u, t) \in T(f)$.*

Définition 2.14. — *Un* morphisme *de systèmes d'approximations de $\mathrm{Apx}_1 : \mathcal{A}_1^{\mathrm{co}} \times \mathcal{L}_1 \to \mathfrak{D}$ à $\mathrm{Apx}_2 : \mathcal{A}_2^{\mathrm{co}} \times \mathcal{L}_2 \to \mathfrak{D}$ est donné par deux foncteurs $F : \mathcal{A}_1 \to \mathcal{A}_2$,*

$G : \mathcal{L}_1 \to \mathcal{L}_2$ tels que

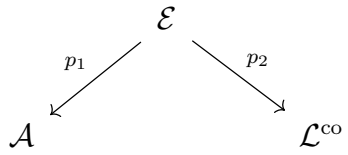


où $\leqslant$ est l'ordre entre les foncteurs introduit tantôt.

Remarque 2.15 (Programmes en tant que suprema de leurs approximations). — *Étant donné un système d'approximations, on peut définir le tiré en arrière*

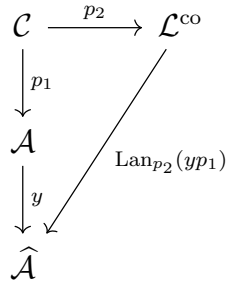
$$\begin{array}{ccc} \mathcal{E}^{\mathrm{co}} & \longrightarrow & \mathbb{1} \\ \downarrow & & \downarrow{\scriptstyle 1} \\ \mathcal{A}^{\mathrm{co}} \times \mathcal{L} & \xrightarrow[\mathrm{Apx}]{} & \mathfrak{D} \end{array}$$

où $\mathbb{1}$ est la catégorie à un objet, et 1 le foncteur envoyant son seul objet sur $\{*\}$ de $\mathfrak{D}$. Il est aisé de voir que $\mathcal{E}^{\mathrm{co}}$ est en fait une sous-catégorie de $\mathcal{A}^{\mathrm{co}} \times \mathcal{L}$. Ainsi, $\mathcal{E}$ est une sous-catégorie de $\mathcal{A} \times \mathcal{L}^{\mathrm{op}}$. En composant l'inclusion avec les deux projections, on obtient ainsi un Span



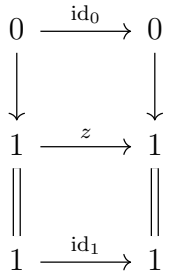
Par la seconde condition de la définition de système d'approximations, p_2 a la propriété de relèvement suivante : dès que l'on a $f : M \to p_2(u)$ dans $\mathcal{L}$, il existe $g : e' \to e$ dans $\mathcal{E}$ tel que $p_2(g) = f$ (notons que le $(-)^{\mathrm{co}}$ renverse l'ordre, pas le sens des morphismes, donc il ne fait aucune différence que l'on considère f dans $\mathcal{L}$ ou $\mathcal{L}^{\mathrm{co}}$).

Remarque 2.16. — *On peut maintenant considérer l'extension de Kan*



qui doit être calculée en respectant l'ordre entre les foncteurs, pas les transformations naturelles (en terme double-catégoriques : ici sont considérées les transformations naturelles verticales*, pas les horizontales, dans le contexte posetal, elles sont dégénérées en ordre précédent). Le fait que $\widehat{\mathcal{A}}$ est complètement dirigée devrait être suffisant pour garantir l'existence de ce type d'extension de Kan, à vérifier.*
Observons de plus que cette construction donne un foncteur de site $\mathcal{L}^{\mathrm{co}}$, non pas $\mathcal{L}$. La signification de cela n'est pas claire. Habituellement, $\mathcal{L}$ est une catégorie, donc $\mathcal{L}^{\mathrm{co}} = \mathcal{L}$, mais en général, il y a une différence et il serait intéressant de voir pourquoi l'ordre a besoin d'être renversé.

Remarque 2.17. — *Si $R : \mathcal{A}^{\mathrm{co}} \times \mathcal{L} \to \mathfrak{D}$ dans Rel, les 2-cellules non-triviales de $\mathfrak{D}$ étant*



Comprendre un foncteur $\mathcal{C}^{\mathrm{co}} \to \mathfrak{D}$ est une sous-catégorie qui est horizontalement un sous-graphe (sous-catégorie même) de $\mathcal{C}^{\mathrm{co}}$, close vers le bas, (i.e.) si $A \in \mathcal{S}$ et $B \leqslant A$ alors $B \in \mathcal{S}$ où $\mathcal{S}$ est ladite sous-catégorie. Si $f : A \to B \in \mathcal{S}$ et $f' : A' \to B' \in \mathcal{S}$, avec $A' \leqslant A$, $B' \leqslant B$ et $f' \leqslant f$, alors $f' \in \mathcal{S}$.

Si $t \sqsubset M$, $t' \sqsubseteq t$, $M \leqslant M'$, alors $t' \sqsubset M'$, et pareil pour les flèches.

$\widehat{R} : \mathcal{L} \to \mathfrak{D}^{\mathcal{A}^{co}} =: \widehat{\mathcal{A}}$, les objets sont les doubles foncteurs entre $\mathcal{A}^{co}$ et $\mathfrak{D}$, les morphismes entre F et G, horizontalement ce sont juste les transformations naturelles habituelles, mais qui représentent des relations. Verticalement, η, c'est des ordres, donc ce sont des inclusions $\eta_t : Ft \subseteq Gt$.

2.2.2 Formalisme bisimulation

LTS et simulations

Définition 2.18. — *Un* système libellé de transitions *(LTS) est un quadruplet $(S, i, \mathscr{L}, \to)$, où S est un ensemble d'états, i l'état initial du système, $\mathscr{L}$ un ensemble de labels, et $\to \subseteq S \times \mathscr{L} \times S$ l'ensemble des transitions libellées.*
Pour simplifier, on note évidemment $(s, a, s') \in \to \Leftrightarrow s \xrightarrow{a} s'$.

Définition 2.19. — *Une* bisimulation *entre deux LTS qui ont le même ensemble de labels $\mathscr{L}$, $(S, i, \mathscr{L}, \to)$ et $(S', i', \mathscr{L}, \to')$, est une relation $B \subseteq S \times S'$ telle que : $(i, i') \in B$ et $\forall (s, s') \in B$,*

- $s \xrightarrow{a} t \Rightarrow \exists t'$, $s' \xrightarrow{a} t'$ *et* $(t, t') \in B$,
- $s' \longrightarrow t' \Rightarrow \exists t$, $s \xrightarrow{a} t$ *et* $(t, t') \in B$.

Remarque 2.20. — *La bisimilarité est plus forte que le fait d'être doublement similaire.*

Bisimulations et λ-calcul

Proposition 2.21. — *La bisimilarité applicative coïncide avec l'équivalence contextuelle : $M \simeq_{ctx} N \Leftrightarrow \forall C$, ($C[M]$ normalise $\Leftrightarrow C[N]$ normalise).*

Remarque 2.22. — *Dans l'appel par nom,* solvable *signifie normalisation de tête, ou de même, arbre de Böhm modulo expansion infinie.*

Définition 2.23. — *Une bisimulation hnf est une relation $B \subseteq \Lambda \times \Lambda$ telle que l'on ait l'un des deux :*

- *ni M ni N n'ont de hnf,*
- *$M \to_h^* \lambda \vec{x}.yM_1 \dots M_m$ et*
  $N \to_h^* \lambda \vec{x}.yN_1 \dots N_n$ et
  $\forall i$, $(M_i, N_i) \in B$.

Exemple 2.24. — $x\langle y\rangle \to (\lambda z.zz)\langle y\rangle \to (\lambda z.zz)\langle I\rangle$
$xy \to (\lambda z.zz)y \to (\lambda z.zz)I \longrightarrow II \to \cdots$.

Remarque 2.25. — *Voir dans la thèse de Morris, $M \xrightarrow{C} M_0$ si et seulement si M_0 hnf, $C[M] \to^* M_0$, et $M \simeq_h M'$ si et seulement si $\forall C$, $C[M]$ une hnf si et seulement si $C[M']$ a une hnf.*

Avec le lemme : si $\approx$ du système ci-dessus, $M \approx M' \Leftrightarrow M \simeq_h M'$.

Digression : applications ouvertes de Joyal

Définition 2.26. — *Soit $\mathcal{C}$ une catégorie supposée pré-topos de Heyting avec un objet d'entiers naturels (le minimum pour interpréter la logique intuitionniste du premier ordre et l'arithmétique).*
*Une classe $\mathcal{S}$ est dite classe d'*applications ouvertes *de $\mathcal{C}$ lorsqu'elle satisfait les axiomes suivants :*

1. *Tout isomorphisme appartient à $\mathcal{S}$, et $\mathcal{S}$ est close par composition.*
2. *(Stabilité) Dans n'importe quel carré cartésien*

$$\begin{array}{ccc} Y' & \longrightarrow & Y \\ {\scriptstyle g}\downarrow & \ulcorner & \downarrow{\scriptstyle f} \\ X' & \xrightarrow[p]{} & X \end{array}$$

   si $f \in \mathcal{S}$, alors g aussi.
3. *(Descente) Dans n'importe quel carré cartésien comme précédemment, si $g \in \mathcal{S}$ et p est épi, alors $f \in \mathcal{S}$.*
4. *Les morphismes $0 \to 1$, et $1 + 1 \to 1$ sont dans $\mathcal{S}$.*
5. *(Sommes) Si $Y \to X$ et $Y' \to X'$ sont dans $\mathcal{S}$, il en va de même pour leur somme $Y + Y' \to X + X'$.*
6. *(Quotients) pour n'importe quel diagramme commutatif*

$$\begin{array}{ccc} Z & \xrightarrow{\quad p \quad} & Y \\ & {\scriptstyle g}\searrow \quad \swarrow{\scriptstyle f} & \\ & B & \end{array}$$

si p est épi et $g \in \mathcal{S}$, alors $f \in \mathcal{S}$.

7. *(Axiome de collection) Pour toute paire $p : Y \twoheadrightarrow X$ et $f : X \longrightarrow A$ où p est épi et $f \in \mathcal{S}$, il existe un carré quasi-cartésien de la forme*

$$\begin{array}{ccc} Z & \longrightarrow & Y \\ \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle f} \\ B & \xrightarrow{\quad h \quad} & A \end{array} \qquad \begin{array}{c} \\ \xrightarrow{\quad p \quad} \\ \\ \twoheadrightarrow \end{array} \begin{array}{c} \\ X \\ \\ A \end{array}$$

où h est épi et $g \in \mathcal{S}$.
(Un tel diagramme est quasi-cartésien lorsque le morphisme évident $Z \longrightarrow B \times_A X$ est un épimorphisme.)

Exemple 2.27. — *Les application continues entre espaces topologiques (?).*

Proposition 2.28. — *Les proto-fibrations dans Cat forment une classe d'applications ouvertes.*

Preuve. Notons $\mathscr{F}$ la classe des proto-fibrations dans Cat.

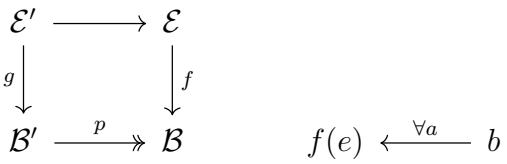
1. Iso $\subseteq \mathscr{F}$ clair. Stabilité par composition :

$$\begin{array}{ccc} \mathcal{E}_1 & & \forall e_1 \\ {\scriptstyle p}\downarrow & & \\ \mathcal{E}_2 & & p(e_1) \xleftarrow{\exists g} e_2 \\ {\scriptstyle q}\downarrow & & \\ \mathcal{E}_3 & & q \circ p(e_1) \xleftarrow{\forall f} b \end{array}$$

   Ok.

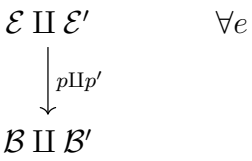
2.

$$\begin{array}{ccccc} \forall e' & & \mathcal{E}' & \xrightarrow{\quad q \quad} & \mathcal{E} \\ & & \downarrow{\scriptstyle g} & \ulcorner & \downarrow{\scriptstyle f} \\ g(e'') \xleftarrow{\forall a} b & & \mathcal{B}' & \xrightarrow{\quad p \quad} & \mathcal{B} \end{array} \qquad q(e) \xleftarrow{\exists} b' \qquad p \circ g(e') = f \circ q(e').$$

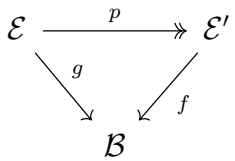
3.


$\exists b', b'' \in \mathcal{B}'$, $f(e) = p(b'')$, $b = p(b')$.
Si $\exists e' \in \mathcal{E}'$ telle que e est sa projection, alors par proto-fibration g, on relève, et on pousse par $\mathcal{E}' \to \mathcal{E}$, et puisque le carré est commutatif, le codomaine de la flèche obtenue est bien e.

4. $0 = \varnothing$ et $1 = \bullet$ est une proto-fibration triviale.
$1 \amalg 1 \to 1$ aussi, puisqu'il n'y a que l'identité à relever.

5. Soient $p, p' \in \mathscr{F}$,


immédiat car $\amalg$ dans Cat « union disjointe ».

6.


p étant épic, g proto-fibration, immédiat, même si p est seulement essentiellement surjectif et plein.

□

Chapitre 3
Sur un calcul de processus

3.1 Introduction

L'*objectif principal de ce chapitre* est de montrer que les différentes formes du théorème de commutation sont des « tirés-en-arrière » d'un théorème de commutation plus général. Nous introduisons à cet effet un calcul de processus correspondant à une forme de logique linéaire différentielle, puis définissons des approximations de Böhm et de Taylor pour ce calcul et prouvons le théorème de commutation à ce niveau de généralité. Nous montrons également que, dès qu'un système $\mathcal{S}$ se plonge dans $\mathcal{P}roc$ de manière suffisamment intéressante, les arbres de Böhm, le développement de Taylor et le théorème de commutation se « tirent-en-arrière » automatiquement à $\mathcal{S}$. Les théorèmes de commutation connus de [27, 29, 44] sont couverts par ces résultats et sont présentés dans le chapitre 4, de même pour les types intersections.

En termes de réseaux de preuves, le calcul $\mathcal{P}roc$ est une représentation du calcul de processus de *structures de preuves* non-typées (*i.e.* , pas nécessairement des objets logiquement corrects) de la logique linéaire différentielle, plus précisément, d'une version non-polarisée des structures de preuve de Honda et de Laurent [34]. La section 4.1.1 donne davantage de détails sur le lien avec les preuves de la logique linéaire. Les règles de réduction de $\mathcal{P}roc$ reflètent le calcul des processus usuel et sont moins fines que celles des réseaux

différentiels habituels [28, 48, 45], tout en étant sémantiquement correctes. Il est à noter que $\mathcal{P}roc$ n'est pas canonique : toute autre syntaxe pour la logique linéaire différentielle classique (par exemple, une extension de la syntaxe d'Accattoli [1]) fonctionnerait probablement. Cependant, l'approche via le calcul $\mathcal{P}roc$ fournit une syntaxe maniable.

Une autre différence est que nous ne considérons pas les sommes formelles. Dans la littérature sur les λ-calculs différentiels, les sommes formelles sont utilisées pour représenter le non-déterminisme : un choix non-déterministe comme $t \to u_1$ et $t \to u_2$ est exprimé par la réduction déterministe $t \to u_1 + u_2$. Comme il est d'usage dans les calculs de processus, il n'y a pas de sommes formelles dans $\mathcal{P}roc$, et le non-déterminisme n'est pas contrôlé (le calcul n'est pas confluent). En outre, ces approximations de Taylor sont *rigides* au sens de [49, 42]. Intuitivement, cela est implicite dans la règle de communication habituelle des calculs de processus polyadiques, qui est quelque chose comme

$$\overline{a}\langle b_1, b_2\rangle \mid a(c_1, c_2).P \to P\{b_1/c_1\}\{b_2/c_2\}$$

(cf. définition 3.17, règle $\otimes/\parr$). Les syntaxes non-rigides comme celles habituellement considérées pour définir les approximations de Taylor correspondraient à b_1, b_2 non-ordonnés dans la sortie $\overline{a}\langle b_1, b_2\rangle$ (i, ils forment un multi-ensemble plutôt qu'une liste) et la réduction ci-dessus deviendrait :

$$\overline{a}\langle b_1, b_2\rangle \mid a(c_1, c_2).P \to P\{b_1/c_1\}\{b_2/c_2\} + P\{b_2/c_1\}\{b_1/c_2\},$$

c'est-à-dire que nous considérons chaque ordre possible du multi-ensemble et utilisons des sommes formelles pour rassembler tous les résultats. L'utilisation de la rigidité et l'abandon des sommes permettent d'importantes simplifications syntaxiques sans aucune perte sémantique. Nous nous restreignons ici en omettant les aspects quantitatifs (les coefficients de la série de Taylor). Cette simplification fait d'autant plus sens que dans un grand nombre de cas (par exemple ceux de [6]), ceux-ci ne sont pas nécessaires.

3.2 Système de réduction

En premier lieu, il convient d'introduire quelques points rapides de vocabulaire tirés des systèmes de réduction utilisés en théorie de la réécriture [47].

Définition 3.1 (ARS). — *Un* système de réduction abstrait $\mathcal{A}$ *est un couple formé d'un ensemble* A *et d'un ensemble de relations binaires* $\to_i$*, indexées par un ensemble* I*,* $\mathcal{A} = (A, \{\to_i \mid i \in I\})$.
Pour $i \in I$*, les relations* $\to_i$ *sont appelées* réductions *ou* relations de réécriture.

Remarque 3.2. — *La définition d'ARS coïncide à celle d'un système de transitions étiqueté (LTS) [46], mais pour faire de la réécriture plutôt que des bisimulations.*

Remarque 3.3 (Calculs et bureaucratie). — *Nous aimerions maintenant définir une notion d'approximation dans les calculs simples tels que le* λ*-calcul et certaines de ses variations, et donc dans les logiques associées. Classiquement, une définition proche de celle de système de réduction avec seulement un type de flèche, comme pour la* β*-réduction, pourrait suffire. Pour des raisons techniques d'encodages dans le chapitre 4, nous introduisons cependant une distinction entre flèche/réduction calculatoire et flèche/réduction administrative. Les réductions administratives sont celles que « l'on doit faire », mais qui ne donnent pas beaucoup d'information du calcul en cours, telles que l'application de règles structurelles en calcul des séquents.*

Définition 3.4 (Système d'approximation). — *Un* système d'approximation *est un ARS* $(A, \{\to_{c_i}, \to_{a_j}\}_{i\in I, j\in J})$ *où*

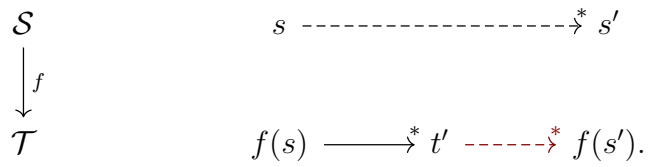
- *les flèches sont étiquetées par les* c_i *sont dites* calculatoires*, et celle étiquetées par les* a_j *sont dite* administrative *;*
- *une* réduction *ou un* chemin *est une suite de flèches ;*
- *une réduction est dite* calculatoire *si elle contient* au moins *une flèche calculatoire.*

Définition 3.5 (Plongement). — *Un* morphisme *entre deux systèmes d'approximation est la donnée d'une application entre les flèches des graphes sous-jacents telle que :*

- *les flèches sont envoyées sur des chemins,*
- *les flèches calculatoires sont envoyées sur des chemins calculatoires.*

Soient $\mathcal{S}$ *et* $\mathcal{T}$ *deux systèmes d'approximation, un* plongement f *entre* $\mathcal{S}$ *et* $\mathcal{T}$ *est un morphisme entre ces deux systèmes et qui reflète les flèches calculatoires :*

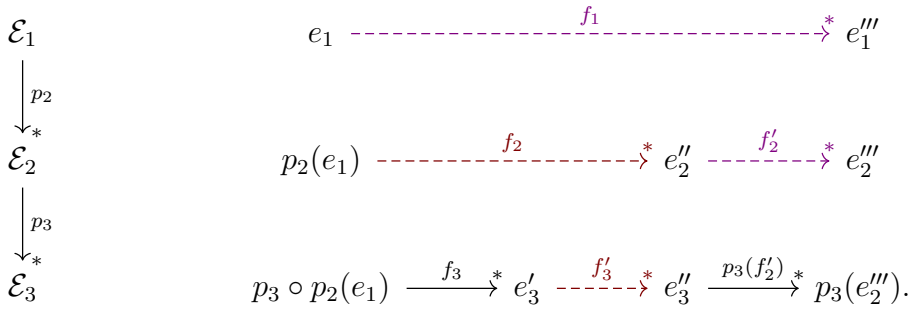
$\forall s \in \mathcal{S}$, $f(s) \to^* t'$ *calculatoire implique* $\exists s' \in \mathcal{S}$, $t' \to^* f(s')$ *et* $s \to^* s'$ *calculatoire. Diagrammatiquement :*



Remarque 3.6 (Catégorique). — *L'intuition issue de la théorie des catégories est ici très présente. En effet, les systèmes de réduction sont moralement vus comme des catégories, les éléments de l'ARS comme les objets, les morphismes comme des foncteurs et les plongements comme des fibrations ou plutôt des* skew-proto-opfibrations. *Plus précisément, le côté* skew *car il faut potentiellement continuer la réduction avant de pouvoir faire le relèvement, et le côté proto car l'unicité n'est pas assurée.*

Lemme 3.7. — *La composition de deux plongements est un plongement.*

Preuve. Prenons deux p_2 et p_3 plongements qui se composent, diagrammatiquement :

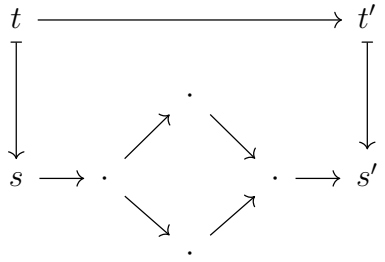


Les couleurs indiquent les étapes. Partons de la réduction f_3 supposée calculatoire, p_3 étant un plongement on en déduit f_2 calculatoire, en utilisant le fait que p_2 est un plongement on en déduit f_1 telle que voulue. □

Remarque 3.8. —

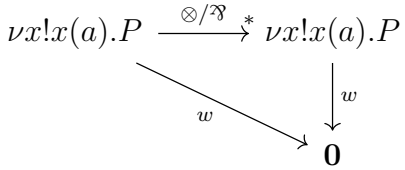
- *Il est possible de se demander « pourquoi pas un simple pré-ordre ou une catégorie ? » pour la définition de système d'approximation.*

Un problème de la "catégorification" de cette notion est la suivante :



en effet, dans ce genre de cas, le plongement aurait à choisir un des chemins de réduction possible et la propriété deviendrait fausse pour l'autre dans tous les cas. Quant aux pré-ordres, ce sont également des catégories.

- *Il est également possible de trouver deux termes tels que l'on peut passer de l'un à l'autre soit via une réduction calculatoire, soit via une réduction administrative. En anticipant les notations le calcul des processus ci-après (section 3.3) :*



dans le cas où l'étape calculatoire se passe dans une boîte que l'on va oublier.

3.3 Processus

Fixons deux ensembles disjoints, infinis et dénombrables de *noms linéaires*, désignés par $a, b, c, \ldots$ et de *noms cartésiens*, désignés par $x, y, z, \ldots$ Comme il est d'usage dans les calculs de processus, nous désignons par $\widetilde{a}$ des suites (éventuellement vides) de noms linéaires, et nous écrivons $|\widetilde{a}|$ pour la longueur de $\widetilde{a}$.

Définition 3.9 (Pré-processus). — *Les* pré-processus *sont définis par la grammaire 3.1.*

$$
\begin{array}{rcll}
P, Q & ::= & \mathbf{0} & \textit{preuve vide} \\
& | & P \mid Q & \textit{mix} \\
& | & a \leftrightarrow b & \textit{axiome} \\
& | & \overline{a}\langle \widetilde{b} \rangle & \textit{tenseur } n\textit{-air} \\
& | & a\langle \widetilde{b} \rangle & \textit{par } n\textit{-air} \\
& & & \\
& | & \overline{a}(x)P & \textit{contraction } n\textit{-air} \\
& | & a(x)P & \textit{cocontraction } n\textit{-air} \\
& | & \nu x P & \textit{coupure exponentielle} \\
& | & \overline{x}\langle a \rangle & \textit{déréliction} \\
& | & x\langle a \rangle & \textit{codéréliction} \\
& | & !x(a).P & \textit{boîte exponentielle}
\end{array}
$$

Grammaire 3.1 : Pré-processus

Les pré-processus linéaires *sont ceux engendrés uniquement par la première partie. Les lettres minuscules* $p, q, \ldots$ *seront utilisés pour désigner les pré-processus linéaires.*

Remarque 3.10. — *Dans la littérature sur les calculs de processus,* $a \leftrightarrow b$ *est généralement appelé* transmetteur linéaire *; nous l'appellerons* axiome *pour souligner le lien avec les réseaux de preuves.*

Définition 3.11 (Contexte). — *Les* contextes *sont définis comme des pré-processus, mais avec l'ajout d'un* trou $\{-\}$. *Comme d'habitude, nous ne considérons que les contextes ayant exactement une occurrence du trou, et nous les désignons par* C. *Nous désignons par* $\mathsf{C}\{P\}$ *le pré-processus obtenu en insérant le pré-processus* P *dans le trou de* C.

Définition 3.12 (Occurrence des noms). — *Un nom linéaire* a *est dit apparaître comme* sujet *dans*

$$\overline{a}\langle \widetilde{b} \rangle, \quad a\langle \widetilde{b} \rangle, \quad \overline{a}(x)P, \quad a(x)P, \quad a \leftrightarrow b, \quad b \leftrightarrow a.$$

Toutes les autres occurrences de a *apparaissent comme* objet.

Les notations νx, $\overline{a}(x)$ *et* $a(x)$ *sont appelées* lieurs cartésiens *: dans* $\nu x P$, $\overline{a}(x)P$ *et* $a(x)P$, *le nom cartésien* x *est lié, et l'*α*-équivalence s'applique comme d'habitude. Un nom cartésien qui n'est pas lié est* libre.

Si un nom cartésien x apparaît ailleurs que dans un lieur, il est dit apparaître comme sujet, *et une telle occurrence est* positive *si elle est de la forme $\overline{x}\langle a\rangle$, ou* négative *si elle est de la forme $x\langle a\rangle$ ou $!x(a).P$.*

Définition 3.13 (Processus). — *Un* processus *(linéaire) est un pré-processus (linéaire) qui vérifie :*

- *Chaque nom linéaire apparaît au maximum deux fois. S'il n'apparaît qu'une fois, il est* libre, *sinon il est* lié *et l'α-équivalence s'applique ;* $\mathsf{fn}(P)$ *désigne l'ensemble des noms libres de P, à la fois linéaires et cartésiens.*
- *Dans $\overline{a}(x)P$ (resp. $a(x)P$) chaque occurrence libre de x dans P (s'il y en a) est positive (resp. négative).*
- *Dans $!x(a).P$, $\mathsf{fn}(P) = \{a\} \cup X$ où X est constitué uniquement de variables cartésiennes ayant des occurrences positives dans P (le cas $X = \emptyset$ est autorisé).*

Définition 3.14 (Congruence structurelle). — *La congruence structurelle est la fermeture réflexive, symétrique, transitive et contextuelle des règles suivantes :*

associativité	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	
commutativité	$P \mid Q \equiv Q \mid P$	
neutre	$P \mid \mathbf{0} \equiv P$	
absorbant	$\nu x\mathbf{0} \equiv \mathbf{0}$	
extrusion de portée	$\beta P \mid Q \equiv \beta(P \mid Q)$	β *lie* $x \notin \mathsf{fn}(Q)$
échange de lieurs	$\beta\gamma P \equiv \gamma\beta P$	β, γ *lieurs cartésiens,*
symétrie	$a \leftrightarrow b \equiv b \leftrightarrow a$	
coupure	$a \leftrightarrow b \mid P \equiv P\{a/b\}$	$b \in \mathsf{fn}(P)$.

Lemme 3.15. — *Être un processus est préservé par la congruence structurelle.*

Preuve. Les règles d'associativité, de commutativité, du neutre, d'échange de lieurs et de symétrie préservent la syntaxe de pré-processus et n'altèrent pas les variables ni leur statut liées/libres, ni leur positivité.

La règle d'absorption élimine ou introduit des variables muettes liées à une coupure exponentielle. La règle d'extrusion de portée prête attention à ne pas capturer de variable.

Pour la règle de la coupure, puisqu'il n'y a qu'une unique occurrence libre de b dans P (car nom linéaire), et qui est substituée avec a dans $P\{a/b\}$, alors la structure de processus est également préservée. □

Remarque 3.16 (sur une traduction). — *Nous avons le « dictionnaire » suivant entre le vocabulaire des calculs de processus et celui des réseaux de preuves de la logique linéaire différentielle :*

Syntaxe	*Calcul des processus*	*Logique Linéaire*
$\mathbf{0}$	*processus terminé*	*preuve vide*
$P \mid Q$	*composition parallèle*	*juxtaposition de réseaux*
$a \leftrightarrow b$	*transmetteur linéaire*	*axiome*
$\overline{a}\langle\widetilde{b}\rangle$	*émission atomique linéaire*	*tenseur n-aire*
$a\langle\widetilde{b}\rangle$	*réception atomique linéaire*	*nœud par n-aire*
$\nu x P$	*restriction de nom*	*coupure exponentielle (*)*
$\overline{a}(x)P$	*préfixe d'émission*	*contraction n-aire (**)*
$a(x)P$	*préfixe de réception*	*cocontraction n-aire (**)*
$\overline{x}\langle a\rangle$	*émission atomique*	*nœud de dérélection*
$x\langle a\rangle$	*réception atomique*	*nœud de codérélict ion*
$!x(a).P$	*émission répliquée (serveur)*	*boîte exponentielle (***)*

(*) : les « coupures linéaires » sont représentées comme expliqué dans la définition 3.18 ci-dessous ;
(**) : l'arité est le nombre d'occurrences de x dans P ;
(***) : avec une porte principale x et autant de portes auxiliaires qu'il n'y a d'occurrences de noms libres dans P.

L'idée est la suivante : les noms libres d'un processus correspondent aux conclusions d'un réseau non-typé. La congruence structurelle correspond à l'égalité des réseaux (au sens des représentations graphiques), enrichie de l'élimination des coupures avec axiomes (ceci est cohérent avec les réseaux d'interaction [28], dans lesquels les axiomes sont juste des fils).

Définition 3.17 (Réduction). — *Les règles de base de la réduction sont les*

suivantes :

$$\begin{array}{rll}
\overline{a}\langle\widetilde{b}\rangle \mid a\langle\widetilde{c}\rangle \mid P & \to_{\otimes/\parr} & P\{\widetilde{b}/\widetilde{c}\} \qquad |\widetilde{b}| = |\widetilde{c}|,\ \widetilde{c} \subseteq \mathsf{fn}(P) \\
\overline{a}(x)P \mid a(x)Q & \to_{!/?} & \nu x(P \mid Q) \\
\nu x(\overline{x}\langle a\rangle \mid x\langle b\rangle \mid P) & \to_{\mathsf{cod}_0} & \nu x P\{a/b\} \qquad b \in \mathsf{fn}(P) \\
\nu x(\mathsf{C}\{\overline{x}\langle a\rangle\} \mid !x(b).P) & \to_{\mathsf{c}} & \nu x(\mathsf{C}\{P\{a/b\}\} \mid !x(b).P) \\
\nu x(!x(a).P \mid Q) & \to_{\mathsf{w}} & \nu x Q \qquad x \notin \mathsf{fn}_+(Q)
\end{array}$$

plus la règle

$$\nu x(!y(c).\mathsf{C}\{\overline{x}\langle a\rangle\} \mid x\langle b\rangle \mid P) \to_{\mathsf{cod}_!} \nu x(!y(c).\nu w\mathsf{C}\{\overline{w}\langle a\rangle\} \mid y\langle c\rangle \mid \nu z(\mathsf{C}\{\overline{z}\langle a\rangle\} \mid z\langle b\rangle) \mid P)$$

dans laquelle w et z sont des nouveaux noms. Dans la règle w, $x \notin \mathsf{fn}_+(Q)$ *signifie que x n'a pas d'occurrence positive libre dans Q.*
Seule la règle $\to_{\otimes/\parr}$ est considérée comme calculatoire pour établir le système de réductions correspondant, les autres règles étant administratives.

Définition 3.18 (Processus sans coupure). —

- *Une* coupure linéaire *dans un processus est un nom linéaire apparaissant deux fois comme sujet.*
- *Une* coupure cartésienne *est un sous-processus de la forme $\nu x P$.*
- *Un processus est* sans coupure *s'il ne contient ni coupure linéaire ni coupure cartésienne.*
- *Un processus P* a une forme sans coupure *si $P \to^* N$ avec N sans coupure.*

Remarque 3.19 (sur DiLL). — *Les règles de base de la définition 3.17 sont une reformulation des étapes d'élimination des coupures des réseaux de preuve de la logique linéaire différentielle.*

- *La règle $\otimes/\parr$ est l'étape multiplicative.*
- *La règle $!/?$ réduit une coupure entre une contraction et une cocontraction. Cependant, au lieu de la règle habituelle $!/?$ qui fait commuter les deux, cette règle crée une coupure cartésienne (qui pourrait également être appeler « coupure exponentielle »).*

- *Les coupures cartésiennes doivent être considérées comme des* zones de communication *au sens d'*Ehrhard *et* Laurent *[26]. Un nombre quelconque de dérélictions et de codérélictions/boîtes (autant que les prémisses de la contraction et de la cocontraction à l'origine de la coupure) peut être apparié de manière non-déterministe dans une coupure cartésienne, en utilisant les règles* cod_0 *(déréliction/codéréliction),* $\mathsf{cod}_!$ *(une codéréliction interagissant avec le bord d'une boîte) ou* c *(une déréliction extrayant une copie d'une boîte).*
- *La règle* w *efface une boîte lorsqu'il n'y a plus de déréliction. Ici, les zones de communication sont considérées comme primitives, plutôt que de les implémenter comme dans [26].*

La présentation habituelle de l'élimination des coupures de la logique linéaire différentielle est basée sur des règles plus fines que celles-ci. Dans leur codage des calculs de processus, Ehrhard *et* Regnier *utilisent cette granularité plus fine pour mettre en œuvre des « zones de communication » garantissant que les entrées peuvent interagir avec les sorties [26]. Cela signifie, qu'ici, cette formulation est sémantiquement correcte par rapport à la formulation habituelle : les règles de granularité plus fine peuvent simuler nos règles. Cependant, notre formulation a l'avantage de correspondre à la tradition des calculs de processus, ainsi que de résoudre les problèmes signalés dans [40]. Ces problèmes sont précisément dus au fait que les règles habituelles d'élimination des coupures sont trop fines pour exprimer la concurrence de la même manière que les calculs de processus.*

Remarque 3.20 (sur les calculs de processus). — *Le calcul de processus* $\mathcal{P}roc$ *présente quelques caractéristiques inhabituelles par rapport aux calculs de processus standards. La plus inhabituelle est la présence de noms linéaires et la convention selon laquelle un nom linéaire est lié dès qu'il apparaît deux fois. Par exemple, le processus* $\overline{a}\langle\rangle \mid a\langle\rangle \mid \overline{b}\langle\rangle$ *serait usuellement écrit* $\nu a(\overline{a}\langle\rangle \mid a\langle\rangle \mid \overline{b}\langle\rangle)$*. Cela conduirait à une prolifération de* ν*, c'est pourquoi il est ici choisi de laisser ces restrictions de noms implicites.*

Une autre différence remarquable est que la réduction à l'intérieur des boîtes est autorisée : si $P \to Q$*, alors* $!x(a).P \to !x(a).Q$*. Ceci est utile pour coder les* λ*-calculs dont les réductions peuvent se produire dans des positions*

arbitraires. C'est essentiel pour récupérer, par exemple, les arbres de Böhm habituels. Dans la section 4.1.2 nous considérons les réductions superficielles*, qui ne se produisent qu'à l'extérieur des boîtes et correspondent à la réduction standard des calculs de processus. C'est la notion de réduction à laquelle les types intersection sont le plus immédiatement applicables. Elle est également utile pour coder les stratégies de réduction faible des λ-calculs, qui ne se réduisent pas sous les λ.*

Remarque 3.21 (sur les coupures). — *Toutes les réductions de base impliquent une coupure. Par conséquent, un processus sans coupure est normal en ce qui concerne la réduction. L'inverse est faux : par exemple, $\overline{a}\langle\rangle \mid a\langle b\rangle$ est un processus normal contenant une coupure avec une « incompatibilité d'arité ». Les coupures auxquelles aucune réduction ne s'applique sont appelées* irréductibles*. D'autres exemples de coupures linéaires irréductibles sont des « conflits » tels que $\overline{a}\langle b\rangle \mid a(x)P$ ou $\overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$, ou des « cercles vicieux » tels que $a \leftrightarrow a$. En logique linéaire : le premier correspond à une coupure entre un tenseur et une contraction ; le second à une coupure entre deux tenseurs ; le troisième à une coupure entre les deux conclusions d'un axiome. Les coupures cartésiennes irréductibles sont de la forme $\nu x(\overline{x}\langle a\rangle \mid P)$ (resp. $\nu x(x\langle a\rangle \mid P)$) avec P ne contenant aucune occurrence négative (resp. positive) de x (en logique linéaire : la première correspond à une coupure entre une déréliction et un affaiblissement, la seconde à une coupure entre une codéréliction et un affaiblissement). Dans la section 4.1.1, seront introduits les types et la correction. Les premiers éliminent les collisions et les incompatibilités d'arité, les seconds les cercles vicieux. En revanche, des coupures cartésiennes irréductibles peuvent être présentes même dans des processus corrects bien typés. Elles correspondent à des situations qui, dans la syntaxe habituelle de la logique linéaire différentielle utilisant des sommes formelles, se réduisent à la somme vide.*

Lemme 3.22. — *$P \to^* Q$ implique $P\{a/b\} \to^* Q\{a/b\}$.*

Démonstration. Par induction sur la longueur de la réduction, nous réduisons au cas d'une étape unique $\mathsf{C}\{R\} \to \mathsf{C}\{R'\}$, cela est prouvé par induction sur C et par la définition 3.17. □

3.4 *Approximations*

3.4.1 Approximations de Taylor

Définition 3.23 (Processus de Taylor). — *Les* processus de Taylor *sont des processus linéaires dans lesquels certaines entrées et sorties sont marquées comme « spéciales » et désignées par* $\overline{a}\langle\!\langle\widetilde{b}\rangle\!\rangle$ *et* $a\langle\!\langle\widetilde{b}\rangle\!\rangle$.
La réduction est définie comme dans les processus linéaires, avec la règle $\otimes/\parr$ *de la définition 3.17, mais elle est restreinte aux paires spécial/spécial et non-spécial/non-spécial, c'est-à-dire que* $\overline{a}\langle\widetilde{b}\rangle \mid a\langle\!\langle\widetilde{c}\rangle\!\rangle$ *est irréductible même dans le cas où* $|\widetilde{b}| = |\widetilde{c}|$.
Le système de réduction ayant comme objets les processus de Taylor est désigné par $\mathcal{T}ay$.

Remarque 3.24 (Oublie). — *Il existe un morphisme évident* $\mathcal{T}ay \to \mathcal{L}in\mathcal{P}roc$ *qui oublie les annotations « spéciales ». Il ne s'agit* pas *d'un plongement :* $\overline{a}\langle\!\langle\rangle\!\rangle \mid a\langle\rangle$ *est envoyé sur* $\overline{a}\langle\rangle \mid a\langle\rangle$, *qui se réduit à* $\mathbf{0}$, *mais le processus original ne peut pas se réduire.*

Lemme 3.25. — *Dans les processus de Taylor, la réduction est fortement confluente et se termine.*

Démonstration. Les seules étapes de réduction sont du type $\otimes/\parr$, et elles ne peuvent pas se superposer, d'où la confluence forte.
Quant à la terminaison, définissons la taille d'un processus de Taylor comme étant le nombre d'axiomes et de sous-processus de la forme $\overline{a}\langle\widetilde{b}\rangle$, $a\langle\widetilde{b}\rangle$, $\overline{a}\langle\!\langle\widetilde{b}\rangle\!\rangle$, $a\langle\!\langle\widetilde{b}\rangle\!\rangle$ qui sont présents dans le processus. En regardant la définition 3.14 et la règle $\otimes/\parr$, nous voyons que la taille est préservée par la congruence structurelle et qu'elle diminue strictement sous l'effet de la réduction, ce qui implique la terminaison. □

Définition 3.26 (Approximations de Taylor). — *La* relation d'approximation de Taylor *est définie par les règles de la figure 3.1. Elle utilise des* jugements d'approximation *de la forme* $p \sqsubset P \vdash \Xi; \Xi'$ *où :*

- p *est un processus de Taylor et* P *un processus arbitraire ;*

$$\frac{}{\mathbf{0} \sqsubset \mathbf{0} \vdash ;} \qquad \frac{}{\overline{a}\langle \widetilde{b} \rangle \sqsubset \overline{a}\langle \widetilde{b} \rangle \vdash ;} \qquad \frac{}{a\langle \widetilde{b} \rangle \sqsubset a\langle \widetilde{b} \rangle \vdash ;} \qquad \frac{}{a \leftrightarrow b \sqsubset a \leftrightarrow b \vdash ;}$$

$$\frac{p \sqsubset P \vdash \Xi ; \Xi' \qquad q \sqsubset Q \vdash \Upsilon ; \Upsilon'}{p \mid q \sqsubset P \mid Q \vdash \Xi, \Upsilon ; \Xi', \Upsilon'} \qquad \frac{p \sqsubset P \vdash \Xi, \widetilde{a} \sqsubset x ; \Xi', \widetilde{b} \sqsubset x}{p\{\widetilde{a}/\widetilde{b}\} \sqsubset \nu x P \vdash \Xi ; \Xi'} \; |\widetilde{a}| = |\widetilde{b}|$$

$$\frac{p \sqsubset P \vdash \Xi, \widetilde{b} \sqsubset x ; \Xi'}{\overline{a}\langle\!\langle \widetilde{b} \rangle\!\rangle \mid p \sqsubset \overline{a}(x)P \vdash \Xi ; \Xi'} \; x \notin \Xi \qquad \frac{p \sqsubset P \vdash \Xi ; \Xi', \widetilde{b} \sqsubset x}{a\langle\!\langle \widetilde{b} \rangle\!\rangle \mid p \sqsubset a(x)P \vdash \Xi ; \Xi'} \; x \notin \Xi'$$

$$\frac{}{a \leftrightarrow b \sqsubset \overline{x}\langle b \rangle \vdash a \sqsubset x ;} \; a \neq b \qquad \frac{}{a \leftrightarrow b \sqsubset x\langle b \rangle \vdash ; a \sqsubset x} \; a \neq b$$

$$\frac{p_1 \sqsubset P\{a_1/a\} \vdash \Xi_1 ; \qquad \cdots \qquad p_n \sqsubset P\{a_n/a\} \vdash \Xi_n ;}{p_1 \mid \cdots \mid p_n \sqsubset !x(a).P \vdash \Xi_1, \ldots, \Xi_n ; a_1 \sqsubset x, \ldots, a_n \sqsubset x} \; \forall i \, a_i \notin \mathsf{fn}(P)$$

Fig. 3.1 : Approximations de Taylor. La notation $\widetilde{a} \sqsubset x$ signifie $a_1 \sqsubset x, \ldots, a_n \sqsubset x$ (où $n = 0$ est possible).

- Ξ *et* Ξ' *sont des ensembles finis disjoints de paires de la forme* $a \sqsubset x$*, où* a *est un nom linéaire n'apparaissant pas librement dans* P*, et* x *un nom cartésien, tel que chaque nom linéaire apparaît au plus une fois dans* $\Xi \cup \Xi'$*.*

Remarque 3.27. — *L'intuition est que* $a \sqsubset x$ *dans* Ξ *(resp.* Ξ'*) signifie que le nom linéaire* a *approxime une occurrence positive (resp. négative) du nom cartésien* x*.*
Dans la suite, nous écrirons $p \sqsubset P$ *lorsqu'un jugement de la forme* $p \sqsubset P \vdash \Xi ; \Xi'$ *est dérivable pour certains* Ξ, Ξ'*.*

Lemme 3.28. — *Soit* $p \sqsubset P$*, alors :*

1. $P = Q\{a/b\}$ *si et seulement s'il existe* $q \sqsubset Q$ *tel que* $p = q\{a/b\}$ *;*
2. *si* $P = \mathsf{C}\{Q\}$ *avec* $\mathsf{fn}(Q) = \{a\} \cup X$ *et* X *est composé de noms cartésiens avec seulement des occurrences positives, alors* $p \equiv t \mid q_1 \mid \cdots \mid q_n$ *pour* q_i *et* t *tels que* $t \sqsubset \mathsf{C}\{\overline{x}\langle a \rangle\}$ *dès que* $x \in \mathsf{fn}(\mathsf{C}\{\overline{x}\langle a \rangle\})$*, et* $q_i \sqsubset Q\{a_i/a\}$ *pour tout* $1 \leqslant i \leqslant n$*, les* a_i *étant deux à deux distincts ;*

3. $P \equiv P'$ *(resp.* $p \equiv p'$*) implique* $p \equiv p'$ *pour un certain* p' *(resp.* $P \equiv P'$ *pour un certain* P'*) tel que* $p' \sqsubset P'$.

Démonstration. Le point (1) est prouvé par induction sur P. Le point (2) est prouvé par induction sur C. Le point (3) est prouvé en vérifiant chaque règle de congruence structurelle (définition 3.14) et ensuite par induction sur les contextes. □

Remarque 3.29. — *Le point (3) du lemme 3.28 assure que nous pouvons utiliser de manière transparente la congruence structurelle avec les approximations de Taylor, ce que nous ferons à partir de maintenant.*

Prouvons maintenant les deux propriétés fondamentales des approximations de Taylor, à savoir qu'elles peuvent être tirées-en-arrière le long de réductions arbitraires et poussées-en-avant le long de réductions à des formes sans coupure.

Lemme 3.30 (Tiré-en-arrière). — *Soit* $P \to^* Q$ *et* $q \sqsubset Q$*, alors, il existe* $p \sqsubset P$ *tel que* $p \to^* q$*. Diagrammatiquement :*

$$\begin{array}{ccc} P & \longrightarrow^* & Q \\ & & \sqcup \\ & & q \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} P & \longrightarrow^* & Q \\ \sqcup & & \sqcup \\ p & \longrightarrow^* & q \end{array}$$

Démonstration. Commençons par prouver le lemme lorsque $P = R$, $Q = R'$ et $R \to R'$ au moyen d'une des règles de réduction de base (définition 3.17).

- Les cas $\otimes/\parr$ et $!/?$ sont des applications du point (1) du lemme 3.28.
- Pour les cas cod_0, $\mathsf{cod}_!$, c et w, on montre que $q \sqsubset R'$ implique $q \sqsubset R$, donc le tiré-en-arrière est la réduction vide. Cette affirmation est directe pour les règles cod_0, $\mathsf{cod}_!$ et w ; pour la règle c, le point (2) du lemme 3.28 est utilisé.

Ensuite, prouvons le lemme pour la réduction en une étape, c'est-à-dire lorsque $P \equiv \mathsf{C}\{R\}$, $Q \equiv \mathsf{C}\{R'\}$ et $R \to R'$ à l'aide d'une règle de base. Définissons la *profondeur* du contexte C comme étant le nombre de boîtes imbriquées à l'intérieur desquelles se trouve le trou. La preuve se fait par induction sur la profondeur de C.

- Si la profondeur est nulle, alors $\mathsf{C} \equiv \nu\widetilde{z}(S \mid \{-\})$ pour un certain S, d'où l'on déduit $q \equiv (s \mid r')\sigma$ où $s \sqsubset S$, $r' \sqsubset R'$ et σ est une substitution.
- Appliquons le résultat que nous avons prouvé ci-dessus, et obtenons $r \sqsubset R$ tel que $r \to^* r'$. Ensuite, prenons $p := (s \mid r)\sigma$, alors $p \sqsubset P$ est tel que, en utilisant le lemme 3.22, $p \to^* q$ comme souhaité. Si C a une profondeur de $d+1$, alors $\mathsf{C} \equiv \nu\widetilde{z}(S \mid !x(a).\mathsf{C}')$ pour un processus S et un contexte C' de profondeur d.
  Nous obtenons donc $q \equiv (s \mid q_1 \mid \cdots \mid q_n)\sigma$ avec $s \sqsubset S$ et $q_i \sqsubset \mathsf{C}'\{R'\}\{a_i/a\}$ pour tout $1 \leqslant i \leqslant n$. Par hypothèse et par le lemme 3.22, nous avons $\mathsf{C}'\{R\}\{a_i/a\} \to \mathsf{C}'\{R'\}\{a_i/a\}$, appliquons donc l'hypothèse d'induction à chaque q_i et obtenons $p_i \sqsubset \mathsf{C}'\{R\}\{a_i/a\}$ tel que $p_i \to^* q_i$. Ensuite, en posant $p := (s \mid p_1 \mid \cdots \mid p_n)\sigma$, nous avons $p \sqsubset P$ tel que $p \to^* q$, et nous concluons.

Enfin, prouvons la version générale du lemme par induction sur la longueur de la réduction $P \to^* Q$. L'énoncé est vrai pour la longueur zéro car $P \equiv Q$. Pour une longueur de $k+1$, nous avons $P \to P_1 \to^* Q$ avec $P_1 \to^* Q$ de longueur k. Étant donné $q \sqsubset Q$, l'hypothèse d'induction donne $p_1 \sqsubset P_1$ tel que $p_1 \to^* q$. Ensuite, appliquons à p_1 le cas prouvé ci-dessus en une étape et nous obtenons le $p \sqsubset P$ souhaité, tel que $p \to^* p_1 \to^* q$. □

Lemme 3.31. — *Soit $p \sqsubset P$ tel que p a une forme sans coupure et $p \to p'$ (réduction en une étape). Il existe alors des réductions $p' \to^* q$ et $P \to Q$ telles que $q \sqsubset Q$. Diagrammatiquement :*

$$\begin{array}{ccc} P & & P \longrightarrow Q \\ \sqcup & \Longrightarrow & \sqcup \quad\quad \sqcup \\ p \longrightarrow p' & & p \longrightarrow p' \longrightarrow^* q \end{array}$$

Démonstration. La preuve se fait par induction sur la structure de P. Nous n'incluons que les cas compliqués, c'est-à-dire lorsque P est une composition parallèle, une boîte ou une restriction.

- Si $P = P_1 \mid P_2$, alors nous savons que $p = p_1 \mid p_2$ avec $p_i \sqsubset P_i$. La réduction peut être effectuée dans l'un des p_i, auquel cas il suffit d'utiliser l'hypothèse d'induction pour conclure, ou il peut s'agir d'une réduction

entre une sortie dans, disons, p_1 et une entrée dans p_2. Via la figure 3.1, cela signifie que P_1 est de la forme $\overline{a}\langle\widetilde{b}\rangle \mid P_1'$ ou $\overline{a}(x)P_1'$, et P_2 de la forme duale, de sorte que P peut se réduire via une communication.

- Si $P = !x(a).P'$, alors $p = p_1 \mid \cdots \mid p_n$ avec $p_i \sqsubset P'\{a_i/a\}$. L'étape $p \to p'$ ne peut pas se produire à cause d'une communication entre deux p_i distincts.

  Si c'était le cas, un nom libre b apparaitrait comme sujet d'entrée dans certains p_i. Or, il n'est pas possible que $b = a_i$, car a_i n'apparaît dans aucun autre p_j avec $j \neq i$. Ainsi, puisque les boîtes ne peuvent pas avoir de nom linéaire libre, b doit approximer un nom cartésien libre de P', mais c'est impossible car les noms cartésiens libres des boîtes ne peuvent apparaître qu'en tant que sorties, et les approximations des sorties sont toujours des sorties, nous obtenons donc $p_j \to p_j'$ pour un certain j, et $p' = p_1 \mid \cdots \mid p_j' \mid \cdots \mid p_n$.

  Comme les approximations n'introduisent pas de coupures, la coupure réduite dans $p_j \to p_j'$ provient d'une coupure de P'. Une telle coupure induit une coupure dans *chaque* p_i, car ces coupures approximent toutes P'. Mais p a une forme sans coupure, donc toutes ces coupures sont réductibles, donc $p_i \to p_i'$ pour *tout* i.

  L'hypothèse d'induction donne alors $p_i' \to^* q_i$ et $P' \to Q_i'$ tels que $q_i \sqsubset Q_i'$. Cependant, comme c'est la même coupure de P' qui est réduite dans chaque $P' \to Q_i'$, tous les Q_i' sont en fait égaux à certains Q', on obtient donc comme souhaité $!x(a).P \to^* !x(a).Q'$ et $q_1 \mid \cdots \mid q_n \sqsubset !x(a).Q'$.

- Si $P = \nu x P_1$, alors $p = p_1\{\widetilde{a}/\widetilde{b}\}$ avec $p_1 \sqsubset P_1$ et $\widetilde{a}, \widetilde{b}$ approximent x comme sortie et entrée, respectivement.

  Si la réduction $p \to p'$ est déjà présente dans p_1, c'est-à-dire si $p' = p_1'\{\widetilde{a}/\widetilde{b}\}$ avec $p_1 \to p_1'$, l'hypothèse d'induction permet de conclure.

  Sinon, la réduction est rendue possible par la substitution de a_i à b_i pour un certain i, cela signifie qu'il existe une occurrence de x en entrée correspondant à une occurrence de x en sortie dans P_1. De telles occurrences sont uniquement déterminées par l'indice i et induisent une réduction en une étape dans P.

Plus précisément, nous avons $P = \nu x \mathsf{C}\{R\}$ et $R \to_{\mathsf{x}} R'$ où x est l'une de cod_0, $\mathsf{cod}_!$ ou c. À ce stade, la preuve se divise en deux cas, selon la forme de C.

- Si C est peu profond (c'est-à-dire que le trou n'est pas sous une boîte), alors nous concluons directement.
- Dans le cas où le trou est sous une boîte, nous utilisons le même argument que ci-dessus (pour le cas $P = !x(a).P'$) pour conclure que l'étape $p \to p'$ peut être « complétée » en réduisant les autres coupures dans p approximativement à la coupure correspondant à $R \to_{\mathsf{x}} R'$, cela donne le poussé-en-avant désiré. □

Lemme 3.32 (Poussé-en-avant). — *Soient $p \sqsubset P$ et $p \to^* n$ tels que n soit sans coupure, alors, il existe une réduction $P \to^* Q$ telle que $n \sqsubset Q$. Diagrammatiquement :*

$$\begin{array}{ccc} P & & P \longrightarrow^* Q \\ \sqcup & \Longrightarrow & \sqcup \quad\quad \sqcup \\ p \longrightarrow^* n & & p \longrightarrow^* n \end{array}$$

Démonstration. Prouvons le résultat suivant, qui est une généralisation du lemme 3.31 aux réductions de longueur arbitraire : pour tout $p \sqsubset P$ avec p ayant une forme sans coupure et pour tout $p \to^* p'$, il existe $p' \to^* q$ et $P \to^* Q$ de sorte que $q \sqsubset Q$. Diagrammatiquement :

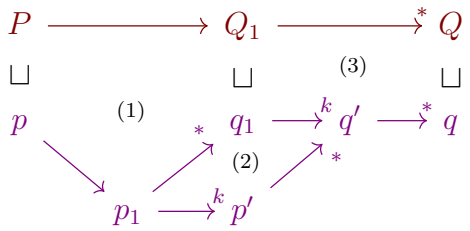
$$\begin{array}{ccc} P & & P \longrightarrow^* Q \\ \sqcup & \Longrightarrow & \sqcup \quad\quad\quad \sqcup \\ p \longrightarrow^* p' & & p \longrightarrow^* p' \longrightarrow^* q \end{array}$$

Le lemme est le cas particulier dans lequel p' est sans coupure, cela implique $q = p'$.

La preuve se fait par induction sur la longueur de la réduction $p \to^* p'$.

- Si la longueur est nulle, alors p est elle-même sans coupure et l'affirmation est immédiate.
- Supposons que la longueur est $k+1$. Cela signifie que $p \to p_1 \to^k p'$, avec $\to^k$ dénotant une réduction en k étapes.

Diagrammatiquement, la preuve peut être représentée comme suit :



où (1) est valable par le lemme 3.31, (2) par le lemme 3.25 (confluence forte) et (3) par l'hypothèse d'induction. En effet,

- par le lemme 3.31, il existe des réductions $p_1 \to^* q_1$ et $P \to^* Q_1$ telles que $q_1 \sqsubset Q_1$. Or, il est bien connu que, dans un système fortement confluent, les deux côtés des diamants de confluence ont la même longueur, c'est-à-dire que si $t \to^m t'$ et $t \to^n t''$, il existe u tel que $t' \to^n u$ et $t'' \to^m u$.
- Par conséquent, et par lemme 3.25, il existe q' tel que $p' \to^* q'$ et $q_1 \to^k q'$. Mais q_1 a toujours une forme sans coupure, car c'est un réduit de p.

L'hypothèse d'induction permet alors de conclure. □

Remarque 3.33. — *Le lemme 3.32 est la raison pour laquelle les entrées et sorties « spéciales » sont requises pour les approximations de Taylor. Si, par exemple $\overline{a}(x)P$ était approximé par $\overline{a}\langle\widetilde{b}\rangle \mid p$ plutôt que $\overline{a}\langle\langle\widetilde{b}\rangle\rangle \mid p$, alors la propriété de poussé-en-avant ne serait pas vérifiée : par exemple, $\overline{a}(x) \mid a\langle\rangle$ ne peut pas se réduire, mais serait approximé par $\overline{a}\langle\rangle \mid a\langle\rangle$, qui lui se réduit à $\mathbf{0}$. Ceci est lié au fait que l'application d'oubli $\mathcal{T}ay \to \mathcal{L}in\mathcal{P}roc$ n'est pas un plongement, comme déjà signalé dans la remarque 3.24.*

3.4.2 Approximations de Böhm et théorème de commutation

Définition 3.34. — *Soient P et Q deux processus de $\mathcal{P}roc$, notons $P \leqslant_0 Q$ lorsque P est obtenu à partir de Q en remplaçant un nombre quelconque de ses boîtes (i.e. de sous-processus de la forme $!x(a).R$) par $\mathbf{0}$.*

La relation d'approximation de Böhm *entre deux processus est la suivante : $N < P$ si*

- *N est sans coupure,*

- *il existe* P' *tel que* $P \to^* P'$ *et* $N \leqslant_0 P'$.

Remarque 3.35. — *Le définition 3.34 s'applique également aux processus de Taylor. Cependant, dans ce cas,* $\leqslant_0$ *dégénère en égalité, car il n'y a pas de boîtes dans les processus de Taylor. Ainsi* $n < p$ *signifie simplement que* n *est la forme sans coupure de* p*, tous les processus de Taylor n'en ayant pas.*

Lemme 3.36. — *Pour tout processus de Taylor sans coupure* n*,* $n \sqsubset P$ *si et seulement s'il existe un processus sans coupure* N *tel que* $n \sqsubset N \leqslant_0 P$.

Démonstration. Formellement, les deux directions se font par induction sur P, ici sont esquissés les points clefs.

Pour le sens direct, la seule façon pour les approximations de Taylor d'« oublier » des coupures est d'approximer une boîte avec $\mathbf{0}$. Donc, si $n \sqsubset P$ avec n sans coupure, soit P est déjà sans coupure, et c'est terminé, soit toutes ses coupures sont à l'intérieur de boîtes approximées par $\mathbf{0}$ dans n. Par conséquent, il suffit de prendre N comme P dans lequel ces mêmes boîtes sont remplacées par $\mathbf{0}$, et nous obtenons N sans coupure tel que $n \sqsubset N \leqslant_0 P$.

La réciproque ne dépend même pas de l'absence de coupure : nous avons $n \sqsubset N$ et N est obtenu à partir de P en remplaçant certaines boîtes par $\mathbf{0}$, mais $\mathbf{0}$ est une approximation de Taylor de n'importe quelle boîte, donc $n \sqsubset P$. □

Théorème 3.37 (de commutation Böhm-Taylor). — *Les relations* $\sqsubset<$ *et* $<\sqsubset$ *coïncident.*

Démonstration. La preuve se trouve dans le diagramme suivant :

$$\begin{array}{ccccc} P & \longrightarrow^* & Q & \geqslant_0 & N \\ \sqcup & (1) & \sqcup & (2) & \sqcup \\ p & \longrightarrow^* & n & = & n \end{array}$$

En effet, par définition $n \sqsubset< P$ est équivalent à la situation décrite dans la partie rouge du diagramme, pour certains Q et N sans coupure.

De même, $n <\sqsubset P$ est équivalent à la situation décrite dans la partie verte du diagramme, pour un certain p.

Pour rouge implique vert, remarquons que les approximations de Taylor n'introduisent pas de coupures, donc N sans coupure implique n sans coupure. Nous obtenons donc le carré (2) par le lemme 3.36, et le lemme 3.30 donne le carré (1).

Pour vert implique rouge, le carré (1) est donné par le lemme 3.32 et le carré (2) par le lemme 3.36. □

3.4.3 Tirer en arrière le théorème de commutation

La conséquence la plus importante du théorème 3.37 est que, dès qu'un système de réduction $\mathcal{S}$ peut être encodé, *plongé* dans $\mathcal{P}roc$, nous disposons d'une notion d'approximations de Böhm et de Taylor pour $\mathcal{S}$ pour laquelle le théorème de commutation habituel s'applique.

Dans ce qui suit, nous fixons un système de réduction arbitraire $\mathcal{S}$ équipé d'un *plongement* $f : \mathcal{S} \to \mathcal{P}roc$, au sens technique de la définition 3.5.

Définition 3.38 (Arbre de Böhm). — *Soit s un objet de $\mathcal{S}$, une* approximation de Böhm *de s est un processus sans coupure N tel que $s \to^* s'$ et $N \leqslant_0 f(s')$.*

*L'*arbre de Böhm *de s, noté $\mathrm{BT}(s)$, est l'ensemble de toutes les approximations de Böhm de s.*

Remarque 3.39. — *Il aurait été possible de définir les approximations de Böhm en demandant que $f(s) \to^* f(s')$ et $N \leqslant_0 f(s')$. Le lemme 3.41 montre que c'est équivalent, car f est un plongement.*

Lemme 3.40. — *Pour tout processus P et processus sans coupure N, $N \leqslant_0 P$ et $P \to^* P'$ implique $N \leqslant_0 P'$.*

Démonstration. Puisque N est sans coupure et que N ne diffère de P que par le fait que certaines boîtes de P sont remplacées par $\mathbf{0}$ dans N, alors toute coupure de P est à l'intérieur d'une boîte. Par conséquent, P' ne diffère de P qu'à l'intérieur de certaines boîtes. En les remplaçant par $\mathbf{0}$, on obtient à nouveau N. □

Lemme 3.41. — *Pour tout objet s de $\mathcal{S}$, N est une approximation de Böhm de s si et seulement si $f(s) \to^* f(s')$ tel que $N \leqslant_0 f(s')$.*

Démonstration. Diagrammatiquement :

$$\begin{array}{ccc} \mathcal{S} & & s \longrightarrow^{*} s' \\ {\scriptstyle f}\downarrow & & \\ \mathcal{P}roc & & f(s) \longrightarrow^{*} f(s') \ {}_{0}\!\geqslant N. \end{array}$$

Pour le sens direct : par définition, nous avons une réduction $s \to^* s'$ dans $\mathcal{S}$ telle que $N \leqslant_0 f(s')$, donc il suffit de prendre la réduction $f(s) \to^* f(s')$ obtenue car f est un morphisme de systèmes d'approximation (autres vocabulaires : par monotonicité, par fonctorialité).

Réciproquement, supposons que $f(s) \to^* f(s')$. Par définition de plongement, il existe s'' tel que $f(s') \to^* f(s'')$ et $s \to^* s''$. Or, par le lemme 3.40, $N \leqslant_0 f(s')$ implique $N \leqslant_0 f(s'')$, donc nous concluons. □

Remarque 3.42. — *La terminologie « arbre de Böhm » est un abus de langage car, à proprement parler,* $\mathrm{BT}(s)$ *n'a rien d'un arbre. Cependant, dans certains cas, il possède les propriétés des arbres de Böhm. Tout d'abord, il découle immédiatement de la définition que* $s \to^* s'$ *implique* $\mathrm{BT}(s') \subseteq \mathrm{BT}(s)$.

Dans le cas où $\mathcal{S}$ *est confluent, c'est-à-dire si toute paire de réductions de la forme* $s_2 {}^*\!\leftarrow s \to^* s_1$ *peut être fermée par une paire de réductions de la forme* $s_1 \to^* s' {}^*\!\leftarrow s_2$*, alors l'implication inverse s'applique également, c'est la proposition 3.43 suivante.*

Proposition 3.43. — *Soit* $\mathcal{S}$ *confluent, alors* $s \to^* s'$ *implique* $\mathrm{BT}(s) = \mathrm{BT}(s')$.

Démonstration. Il suffit de montrer que $\mathrm{BT}(s) \subseteq \mathrm{BT}(s')$. Soit $N \in \mathrm{BT}(s)$. Par définition, $s \to^* s_1$ tels que $N \leqslant_0 f(s_1)$. Par confluence, il existe s'' tel que $s' \to^* s''$ et $s_1 \to^* s''$, donc la conclusion suit par lemme 3.40. □

Remarque 3.44. — *En outre, lorsque* $\mathcal{S}$ *est confluent,* $\mathrm{BT}(s)$ *peut être considéré comme un processus sans coupure éventuellement infini, au sens où l'on prend la construction* $!x(a).P$ *de manière coinductive dans le lemme 3.45 suivant.*

Lemme 3.45. — *Soit* $\mathcal{S}$ *confluent, alors* $\mathrm{BT}(s)$ *est soit vide, soit un idéal par rapport à* $\leqslant_0$.

Démonstration. La fermeture vers le bas est immédiate du fait de la définition et de la transitivité de $\leqslant_0$. Il reste à prouver que $N_1, N_2 \in \mathrm{BT}(s)$ implique qu'il existe $N \in \mathrm{BT}(s)$ tel que $N_1, N_2 \leqslant_0 N$.

Commençons par prouver que, étant donné un processus arbitraire P, le poset $\{Q \mid Q \leqslant_0 P\}$ ordonné par $\leqslant_0$ possède un suprema binaire.

Soit τ_P la forêt enracinée dont les nœuds sont les boîtes de P et telle qu'il existe une arête de R vers S si S est un sous-processus de R. Par définition, $Q \leqslant_0 P$ si Q est obtenu en remplaçant certaines boîtes de P par $\mathbf{0}$, de sorte que le poset $\{Q \mid Q \leqslant_0 P\}$ est isomorphe au poset $\{\tau \mid \tau \text{ est une forêt enracinée de } \tau_P\}$ ordonné par inclusion enracinée de forêts, et ce dernier possède évidemment un suprema binaire.

Maintenant, par définition, $N_1, N_2 \in \mathrm{BT}(s)$ signifie que $s \to^* s_1$ et $s \to^* s_2$ tels que $N_1 \leqslant_0 f(s_1)$ et $N_2 \leqslant_0 f(s_2)$. Par confluence, on a s' tel que $s_1 \to^* s'$ et $s_2 \to^* s'$. Par le lemme 3.40 et le fait que f est un morphisme, $N_1, N_2 \leqslant_0 f(s')$, prenons donc N comme étant le supremum de N_1 et N_2. □

Définition 3.46 (Développement de Taylor). — *Soit s un objet de $\mathcal{S}$. Une* approximation de Taylor *de s est un processus de Taylor p tel que $p \sqsubset f(s)$. Le* Développement de Taylor *de s, noté $\mathcal{T}(s)$, est l'ensemble de toutes les approximations de Taylor de s.*

Le déploiement de Taylor de l'arbre de Böhm *de s est l'ensemble suivant de processus de Taylor :*

$$\mathcal{T}(\mathrm{BT}(s)) := \{n \sqsubset N \mid N \in \mathrm{BT}(s)\}.$$

Remarquez que, puisque les approximations de Böhm sont sans coupure, $\mathcal{T}(\mathrm{BT}(s))$ est en fait un ensemble de processus linéaires sans coupure.

Définition 3.47. — *Soit X un ensemble arbitraire de processus de Taylor, $\mathrm{NF}(X)$ désigne l'ensemble des formes sans coupure des processus dans X :*

$$\mathrm{NF}(X) := \{n \textit{ sans coupure} \mid \exists p \in X \textit{ tel que } p \to^* n\}.$$

Théorème 3.48 (de commutation Böhm-Taylor, tiré en arrière). — *Pour tout objet s de $\mathcal{S}$,*

$$\mathrm{NF}(\mathcal{T}(s)) = \mathcal{T}(\mathrm{BT}(s)).$$

Démonstration. En déroulant les définitions, nous avons $n \in \mathrm{NF}(\mathcal{T}(s))$ si $n <\sqsubset f(s)$. De même, $n \in \mathcal{T}(\mathrm{BT}(s))$ si $n \sqsubset< f(s)$: l'implication directe est immédiate à partir des définitions et du fait que f est un morphisme ; la réciproque découle du lemme 3.41, concluons alors par le théorème 3.37. □

Chapitre 4
Sur des applications

Nous nous appliquons ici à utiliser la théorie précédemment construite. Dans section 4.1 sont développé les notions de correction et de typage pour le calcul de processus $\mathcal{P}roc$, puis les types intersections qui peuvent également se faire « tirer-en-arrière » le long de certains plongements. Les sections suivantes développent des encodages du $\lambda!$-calcul, du $\lambda\mu$-calcul, ainsi que d'une variante polyadique hyperlocalisée du π-calcul dans $\mathcal{P}roc$.

4.1 *Logique linéaire*

4.1.1 Preuves comme Processus

Le calcul des séquents de la logique linéaire classique est présenté dans la figure 4.1, dans lequel sont omis les connecteurs additifs car non-essentiels à notre propos. Les séquents sont divisés en trois parties, pour correspondre à notre calcul de processus. La proposition 4.1 présentation est équivalente à une présentation plus standard, telle que celle de Girard.

Proposition 4.1. — *Un séquent* $\vdash \Theta; \Theta'; \Gamma$ *est prouvable dans le calcul des séquents de la figure 4.1 si le séquent* $\vdash ?\Theta, !\Theta', \Gamma$ *est prouvable dans le calcul des séquents de la logique linéaire classique donné dans [32].*

Démonstration. Observons que la traduction de chaque règle de la figure 4.1

$$\frac{}{\vdash ; ; A^\perp, A} \qquad \frac{\vdash \Theta_1; \Theta'_1; \Gamma, A^\perp \qquad \vdash \Theta_2; \Theta'_2; \Delta, A}{\vdash \Theta_1, \Theta_2; \Theta'_1, \Theta'_2; \Gamma, \Delta}$$

$$\frac{\vdash \Theta_1, A^\perp, \dots, A^\perp; \Theta'_1; \Gamma \qquad \vdash \Theta_2; \Theta'_2, A; \Delta}{\vdash \Theta_1, \Theta_2; \Theta'_1, \Theta'_2; \Gamma, \Delta} \qquad \frac{\vdash \Theta; \Theta'; \Gamma, A_1, \dots, A_n}{\vdash \Theta; \Theta'; \Gamma, A_1 \parr \cdots \parr A_n}$$

$$\frac{\vdash \Theta_1; \Theta'_1; \Gamma_1, A_1 \qquad \cdots \qquad \vdash \Theta_1; \Theta'_n; \Gamma_n, A_n}{\vdash \Theta_1, \dots, \Theta_n; \Theta'_1, \dots, \Theta'_n; \Gamma_1, \dots, \Gamma_n, A_1 \otimes \cdots \otimes A_n}$$

$$\frac{\vdash \Theta; \Theta'; \Gamma, A}{\vdash \Theta, A; \Theta'; \Gamma} \qquad \frac{\vdash \Theta, A, \dots, A; \Theta'; \Gamma}{\vdash \Theta; \Theta'; \Gamma, ?A} \qquad \frac{\vdash \Theta; ; A}{\vdash \Theta; A;} \qquad \frac{\vdash \Theta; \Theta', A; \Gamma}{\vdash \Theta; \Theta'; \Gamma, !A}$$

Fig. 4.1 : Logique linéaire. Les règles d'échange (applicables à chaque segment de séquent) sont implicites.

est dérivable dans le calcul de Girard. Réciproquement, chaque règle du calcul de Girard est dérivable dans celui-ci. La dérivation de la contraction et de la promotion introduit des coupures, en utilisant la dérivabilité de $\vdash A; ; !A^\perp$.

Les règles de contraction et de promotion sont plus subtiles, elles nécessitent d'introduire des coupures. Pour la contraction, en commençant avec $\vdash ; ; \Gamma, ?A, ?A$ et, en exploitant la dérivabilité de $\vdash A; ; !A^\perp$, nous obtenons $\vdash A, A; ; \Gamma$ via deux coupures, à partir desquelles se dérive $\vdash ; ; \Gamma, ?A$. Pour la promotion, il suffit de procéder de même. □

Remarque 4.2. — *Le calcul des séquents de la figure 4.1 peut être décoré avec des processus et converti en un système de types. Ceci correspond à une présentation « à la Curry » de la correspondance entre nos processus et la logique linéaire classique. Nous optons plutôt pour une présentation « à la Church » dans la définition 4.3 suivante.*

Définition 4.3 (Processus typé). — *Un* processus typé *est un processus dans lequel chaque occurrence de nom (sauf dans les lieurs) est décorée par une formule de la logique linéaire, de telle sorte que :*

- *dans* $a^A \leftrightarrow b^B$, $B = A^\perp$;

- *deux occurrences du même nom linéaire sont décorées par la même formule si l'une est sujet et l'autre objet, ou par des formules doubles si elles sont toutes les deux sujet ou toutes les deux objet ;*
- *dans* $\overline{a}^A\langle b_1^{B_1}, \dots, b_n^{B_n}\rangle$ *(resp.* $a^A\langle b_1^{B_1}, \dots, b_n^{B_n}\rangle$*),* $A = B_1 \otimes \cdots \otimes B_n$ *(resp.* $A = B_1 \parr \cdots \parr B_n$*);*
- *toutes les occurrences de même polarité d'un nom cartésien sont décorées par la même formule ;*
- *dans* $\nu x P$*, les occurrences positives et négatives de* x *(le cas échéant) sont décorées par des formules duales ;*
- *dans* $\overline{a}^A(x)P$ *(resp.* $a^A(x)P$*),* $A = ?B$ *(resp.* $A = !B$*), où* B *est la formule décorant* x *dans* P*, ou est arbitraire si* $x \notin \mathsf{fn}(P)$ *;*
- *dans* $\overline{x}^A\langle a^B\rangle$ *et* $x^A\langle a^B\rangle$*, nous avons* $A = B$ *; de même, dans* $!x^A(a).P$*, la décoration de* a *dans* P *(qui doit apparaître) est* A*.*

Le type *d'une occurrence libre de nom comme sujet (resp. comme objet) est sa décoration (resp. la négation de sa décoration).*

Le séquent $\vdash \Theta; \Theta'; \Gamma$ *est dit* associé à P *lorsque* Θ *(resp.* Θ'*,* Γ*) contient tous les types de toutes les occurrences libres cartésiennes positives (resp. négatives, cartésiennes, linéaires) de variables. Ceci est unique à une permutation des occurrences près. Dans la suite nous parlerons du « séquent » associé à* P*.*

Remarque 4.4 (Sur la correction et le typage). — *Il est important de noter que le fait d'être typé n'implique pas d'être logiquement correct. Par exemple,* $a^A \leftrightarrow a^{A^\perp}$ *est typé, mais le séquent qui lui est associé est vide. Dans la littérature sur la logique linéaire, il existe des* critères de correction *[32, 18] pour isoler les objets « semblables à des preuves ». Ici la présentation dans la figure 4.2 est inductive. Un jugement* $P \rhd \Xi; \Xi'; \Gamma$ *signifie que* P *est correct et que ses noms cartésiens positifs libres (resp. cartésiens négatifs, linéaires) sont dans* Ξ *(resp.* Ξ'*,* Γ*). De plus, suivant les usages en logique linéaire, le typage et la correction sont des notions indépendantes : un processus correct n'est pas nécessairement typable, et* vice versa.

$$\frac{a \neq b}{a \leftrightarrow b \rhd \Xi; ; a, b} \qquad \frac{P \rhd \Xi; \Xi'_1; \Gamma, a \qquad Q \rhd \Xi; \Xi'_2; \Delta, a}{P \mid Q \rhd \Xi; \Xi'_1, \Xi'_2; \Gamma, \Delta}$$

$$\frac{}{\overline{a}\langle \widetilde{b} \rangle \rhd \Xi; ; a, \widetilde{b}} \qquad \frac{P \rhd \Xi; \Xi'; \Gamma, \widetilde{b}}{a\langle \widetilde{b} \rangle \mid P \rhd \Xi; \Xi'; \Gamma, a}$$

$$\frac{P \rhd \Xi, x; \Xi'_1; \Gamma \qquad Q \rhd \Xi; \Xi'_2, x; \Delta}{\nu x(P \mid Q) \rhd \Xi; \Xi'_1, \Xi'_2; \Gamma, \Delta} \qquad \frac{}{\overline{x}\langle a \rangle \rhd \Xi, x; ; a}$$

$$\frac{P \rhd \Xi, x; \Xi'; \Gamma}{\overline{a}(x)P \rhd \Xi; \Xi'; \Gamma, a} \qquad \frac{P \rhd \Xi; ; a}{!x(a).P \rhd \Xi; x;} \qquad \frac{P \rhd \Xi; \Xi', x; \Gamma}{a(x)P \rhd \Xi; \Xi'; \Gamma, a}$$

Fig. 4.2 : Processus corrects.

Proposition 4.5. — *Un séquent est prouvable dans le calcul des séquents de la figure 4.1 s'il est associé à un processus typé correct. De plus, les preuves sans coupure correspondent à des processus sans coupure.*

Démonstration. Les deux directions se font par induction, sur la dernière règle de la preuve du calcul des séquents ou sur la dernière règle de la dérivation de la correction. □

Remarque 4.6. — *Les processus typés corrects peuvent donc être considérés comme des preuves de la logique linéaire, et la réduction comme une procédure d'élimination des coupures. La question de la confluence et de la terminaison se pose. Elles doivent pouvoir être prouvées pour les processus typés corrects, cela pourra faire l'objet de travaux futurs. Observons cependant que la figure 4.2 peut facilement être étendu pour inclure la correction des processus non-déterministes. Il suffit de remplacer la dernière règle de la figure 4.2 par la règle ci-dessous à gauche, et d'ajouter la règle ci-dessous à droite :*

$$\frac{P_1 \rhd \Xi; \Xi'_1, x; \Gamma_1 \qquad \cdots \qquad P_n \rhd \Xi; \Xi'_n, x; \Gamma_n}{a(x)(P_1 \mid \cdots \mid P_n) \rhd \Xi; \Xi'_1, \ldots, \Xi'_n; \Gamma_1, \ldots, \Gamma_n, a} \qquad \frac{}{x\langle a \rangle \rhd \Xi; x; a}$$

Les processus typés corrects (pour cette notion étendue de correction) correspondent aux preuves de la logique linéaire différentielle (avec promotion mais sans zéro ni somme).

Remarque 4.7 (Correction des règles de mélange). — *Il est également possible d'ajouter la correction de ce que l'on appelle les* règles de mélange *: nous ajoutons une règle nullaire dérivant* $\mathbf{0} \rhd ;;$ *et une règle binaire dérivant* $P_1 \mid P_2 \rhd ; \Xi'_1, \Xi'_2; \Gamma_1, \Gamma_2$ *de* $P_i \rhd ; \Xi'_i; \Gamma_i$*,* $i \in \{1, 2\}$ *:*

$$\frac{}{\mathbf{0} \rhd ;;} \qquad \frac{P \rhd \Xi; \Xi'_1; \Gamma \qquad Q \rhd \Xi; \Xi'_2; \Delta}{P \mid Q \rhd \Xi; \Xi'_1, \Xi'_2; \Gamma, \Delta}$$

Cette notion plus générale de correction, pour les processus non déterministes et avec des règles de mélange, est celle que nous examinerons dans la section suivante.

4.1.2 Types intersection

Les types intersection [16, 8], en particulier dans leur version non-idempotente [31, 13], sont liés aux approximations de Taylor [14, 22, 42].

Un système de types intersection non-idempotents est ici fourni pour $\mathcal{P}roc$ et nous montrons comment ce système se « tire-en-arrière » le long des plongements, caractérisant automatiquement l'existence de formes normales via l'inférence de types, tant que celles-ci sont correctement reflétées dans $\mathcal{P}roc$.

Les exemples connus, tels que ceux cités ci-dessus ou ceux de [11], s'inscrivent dans ce cadre. D'après [42], nous savons que d'autres formes de types intersection (affine, idempotente) peuvent être traitées de manière similaire. Nous nous limitons ici au cas linéaire non idempotent.

Définition 4.8. —

- *Les* types intersections *sont définis dans la grammaire 4.1. La dualité (i.e. négation linéaire) est définie de manière usuelle, avec* $\wedge$ *dual à* $\vee$*.*
- *Un* jugement de type d'intersection *est de la forme* $P \vdash \Xi; \Xi'; \Gamma$ *où* Ξ *et* Ξ' *(resp.* Γ*) contiennent des déclarations de type de la forme* $x : A$ *(resp.* $a : A$*) avec* x *un nom cartésien (resp.* a *un nom linéaire) et* A *un type intersection. Le même nom cartésien peut apparaître dans plusieurs déclarations de type dans* Ξ *et* Ξ'*, même plusieurs fois avec le même type (cela signifie que nous considérons des types intersection non idempotents). En revanche, un nom linéaire ne peut être déclaré qu'une seule fois dans*

$$\begin{array}{rcl} A, B & ::= & X \\ & | & X^{\perp} \\ & | & A_1 \otimes \cdots \otimes A_n \\ & | & A_1 \parr \cdots \parr A_n \\ & | & A_1 \wedge \cdots \wedge A_n \\ & | & A_1 \vee \cdots \vee A_n. \end{array}$$

Grammaire 4.1 : Types intersections.

Γ. *Le système de types intersection pour les processus est donné dans la figure 4.3.*

Définition 4.9. — *Un* processus de Taylor typé *est défini comme dans la définition 4.3, avec l'ajout des contraintes que dans* $\overline{a}^{A}\langle\!\langle b_1^{B_1}, \ldots, b_n^{B_n}\rangle\!\rangle$ *(resp.* $a^{A}\langle\!\langle b_1^{B_1}, \ldots, b_n^{B_n}\rangle\!\rangle$*) nous avons* $A = B_1 \cdots \vee B_n$ *(resp.* $A = B_1 \wedge \cdots \wedge B_n$*).*

Remarque 4.10. — *Les processus de Taylor n'ont pas de noms cartésiens, de sorte que le séquent associé à un processus de Taylor typé est de la forme* $\vdash ; ; \Gamma$*, que nous écrivons simplement* $\vdash \Gamma$*. Si* p *est un processus de Taylor typé, nous écrivons* p^{-} *pour le processus sous-jacent, sans les décorations. Dans ce qui suit, la correction est entendue au sens généralisé de la fin de la section 4.1.1 (avec mélange).*

Lemme 4.11. — *Si* p *est un processus de Taylor typé correct, alors* p^{-} *a une forme sans coupure.*

Démonstration. Les processus de Taylor sont linéaires, donc la réduction se termine toujours (lemme 3.25). Il suffit alors de montrer qu'il n'y a pas de coupures irréductibles. En effet, les coupures irréductibles sont incorrectes ou non typables et la réduction préserve la correction et les décorations de type. □

Remarque 4.12. — *La suite est une reformulation des résultats de [42], où un lien générale entre les approximations de Taylor et les types intersection est détaillé.*

$$\frac{}{\mathbf{0} \vdash ; ;} \qquad \frac{P \vdash \Xi_1; \Xi_1'; \Gamma, [a : A] \quad Q \vdash \Xi_2; \Xi_2'; \Delta, [a : A^\perp]}{P \mid Q \vdash \Xi_1, \Xi_2; \Xi_1', \Xi_2'; \Gamma, \Delta} \qquad \frac{}{a \leftrightarrow b \vdash ; ; a : A, b : A^\perp}$$

$$\frac{}{\overline{a}\langle b_1, \ldots, b_n\rangle \vdash ; ; a : \bigotimes_i A_i, b_1 : A_1^\perp \ldots, b_n : A_n^\perp}$$

$$\frac{}{a\langle b_1, \ldots, b_n\rangle \vdash ; ; a : \parr_i A_i, b_1 : A_1^\perp \ldots, b_n : A_n^\perp}$$

$$\frac{P \vdash \Xi_1, x : A_1, \ldots, x : A_n; \Xi_1'; \Gamma \quad Q \vdash \Xi_2; \Xi_2', x : A_1^\perp, \ldots, x : A_n^\perp; \Delta}{\nu x(P \mid Q) \vdash \Xi_1, \Xi_2; \Xi_1', \Xi_2'; \Gamma, \Delta}$$

$$\frac{P \vdash \Xi, x : A_1, \ldots, x : A_n; \Xi'; \Gamma}{\overline{a}(x)P \vdash \Xi; \Xi'; \Gamma, a : \bigvee_i A_i}$$

$$\frac{P_1 \vdash \Xi_1; \Xi_1', x : A_1; \Gamma_1 \quad \ldots \quad P_n \vdash \Xi_n; \Xi_n', x : A_n; \Gamma_n}{a(x)(P_1 \mid \cdots \mid P_n) \vdash \Xi_1, \ldots, \Xi_n; \Xi_1', \ldots, \Xi_n'; \Gamma_1, \ldots, \Gamma_n, a : \bigwedge_i A_i}$$

$$\frac{}{\overline{x}\langle a\rangle \vdash x : A; ; a : A^\perp} \qquad \frac{}{x\langle a\rangle \vdash ; x : A; a : A^\perp}$$

$$\frac{P \vdash \Xi_1; ; a : A_1 \quad \ldots \quad P \vdash \Xi_n; ; a : A_n}{!x(a).P \vdash \Xi_1, \ldots, \Xi_n; x : A_1, \ldots, x : A_n;}$$

Fig. 4.3 : Types intersection. Dans la deuxième règle du haut, les déclarations $a : A$ et $a : A^\perp$ sont soit toutes les deux présentes, soit toutes les deux absentes (la règle est une coupure dans le premier cas, un mélange dans le second).

Proposition 4.13. — *Le jugement* $P \vdash \Xi; \Xi'; \Gamma$ *est dérivable dans le système de la figure 4.3 s'il existe un processus de* Taylor *typé correct* p *dont le séquent associé est* $\vdash \Xi, \Xi', \Gamma$ *tel que* $p^- \sqsubset P$.

Esquisse de preuve. Intuitivement, le résultat suit de l'observation que la figure 4.3 est une superposition des figure 3.1 et figure 4.2 (avec les règles supplémentaires données à la fin de la section 4.1.1), annotée avec des types. □

Lemme 4.14 (Expansion du sujet). — *Si* $Q \vdash ; \Xi'; \Gamma$ *est dérivable et* P *est correct de sorte que* $P \to Q$*, alors* $P \vdash ; \Xi'; \Xi'; \Gamma$ *est dérivable.*

Démonstration. Le résultat est une reformulation du lemme 3.30 à l'aide de la proposition 4.13, il faut néanmoins ajouter des décorations de type. La correction de P garantit que l'approximation tirée-en-arrière est également correcte. □

Définition 4.15 (Superficialité). —

- *Un contexte est dit* superficiel *si le trou n'apparaît pas à l'intérieur d'une boîte.*
- *Une réduction superficielle est notée* $\to_0$.
- *Un processus est dit* sans coupure superficielle *lorsque toutes ses coupures, s'il y en a, sont à l'intérieur de boîtes.*

Définition 4.16 (Réduction peu profonde). — *Une réduction est dite* peu profonde *lorsqu'elle suit la définition 3.17 modifiée comme suit : la règle de réduction* $\mathsf{cod}_!$ *est supprimée,* c *est restreint au cas* $\mathsf{C} = Q \mid \{-\}$*, avec* Q *arbitraire. Les règles de réduction ne sont fermées que pour les contextes peu profonds.*

Lemme 4.17 (Progrès). — *Soit* $P \vdash \Xi; \Xi'; \Gamma$ *est dérivable et soit* p *est le processus de Taylor typé associé selon la proposition 4.13. Alors, soit* P *est sans coupure superficielle, soit* $P \to_0^* Q$ *et il existe une dérivation* $Q \vdash \Xi; \Xi'; \Gamma$ *dont le processus de Taylor associé* q *est tel que* $p^- \to q^-$.

Démonstration. En observant la figure 3.1, nous voyons qu'une coupure peu profonde dans P produit une coupure dans p. Comme observé dans le lemme 4.11, une telle coupure ne peut pas être irréductible, donc nous avons $p \to q$ en la réduisant. En supprimant les annotations de type, nous avons $p^- \sqsubset P$ et $p^- \to q^-$, nous appliquons donc les lemme 3.32 et proposition 4.13. □

Remarque 4.18. — *Le lemme 4.17 ne s'applique pas aux coupures générales : puisque les boîtes peuvent être approximées par* $\mathbf{0}$*,* P *peut contenir une coupure à l'intérieur d'une boîte qui est invisible pour la dérivation du type d'intersection.*

Théorème 4.19. — *Un processus P est typable comme dans la figure 4.3 s'il est correct (au sens généralisé de la fin de la section 4.1.1) et $P \to_0^* P_0$ avec P_0 superficiel et sans coupure.*

Démonstration. Soit P un processus typable. La correction est immédiate à partir des règles de typage : elles sont essentiellement une décoration de la figure 4.2 plus les règles de correction supplémentaires à la fin de la section 4.1.1. Nous devons montrer que P se réduit à un processus sans coupure peu profond.

Soit p l'approximation de Taylor typée donnée par la proposition 4.13. Nous raisonnons par induction sur la taille de p^-, telle que définie dans la preuve du lemme 3.25.

Nous appliquons le lemme 4.17 et concluons immédiatement car P est sans coupure superficielle ou obtenons $P \to^* Q$ avec Q typable avec une approximation associée q telle que $p^- \to q^-$. Cela implique que la taille de q^- est strictement plus petite que la taille de p^-, donc nous concluons par l'hypothèse d'induction.

Supposons maintenant que $P \to_0^* P_0$ avec P_0 sans coupure peu profonde. Il est facile de prouver, par induction sur P_0, que P_0 est typable. Intuitivement, nous approximons toutes les boîtes par 0 et le typage est alors garanti par l'exactitude et l'absence de coupures. Nous concluons par le lemme 4.14. □

Remarque 4.20. — *Si $f : \mathcal{S} \to \mathcal{P}roc$ est un morphisme de systèmes de réduction, alors nous pouvons dire qu'un objet de s est* typable dans les types intersection *si $f(s)$ est typable selon la figure 4.3.*

Définition 4.21. — *Soit $\mathcal{P}roc_0$ le système de réduction avec les processus comme objets mais avec $\to_0^*$ comme réductions. Dans ce qui suit, nous considérons un système de réduction $\mathcal{S}$ avec un ensemble distingué d'objets appelé de manière suggestive « normal ». Un plongement $f : \mathcal{S} \to \mathcal{P}roc_0$ est dit* consistant *lorsque*

- *pour tout objet s, $f(s)$ est correct (au sens généralisé);*
- *pour tout objet s_0, s_0 est normal si $f(s_0)$ est sans coupure superficielle.*

Théorème 4.22. — *Soit $f : \mathcal{S} \to \mathcal{P}roc_0$ un plongement consistant, alors, un objet s de $\mathcal{S}$ est typable dans les types intersection si $s \to^* s_0$ avec s_0 normal.*

Démonstration. Supposons que s est typable, ce qui signifie que $f(s)$ l'est. Par le théorème 4.19, $f(s) \to_0^* P_0$ avec P_0 superficiel et sans coupure. Puisque f est un plongement, nous avons $P_0 \to_0^* f(s_0)$ tel que $s \to_0^* s_0$. Mais P_0 est sans coupure superficielle, donc $f(s_0) = P_0$ et nous concluons que s_0 est normal par consistance.

Supposons à présent que $s \to^* s_0$ avec s_0 normal, ce qui implique $f(s) \to_0^* f(s_0)$. Par consistance, $f(s_0)$ est correct et sans coupure superficielle, il est donc typable. Par consistance et par le lemme 4.14, $f(s)$ est typable, donc s est typable par définition. □

4.2 *Call-by-push-value*

Il est connu que l'appel-par-valeur de Paul Levy [37] peut être exprimé en logique linéaire intuitionniste [21], ce qui donne le *bang-calcul* [25], ou $\lambda!$-calcul. Ici est utilisée une reformulation récente due à Bucciarelli *et al.* [11], qui permet de montrer en même temps comment les substitutions explicites d'Accattoli et Kesner « à distance » peuvent être manipulées sans heurts.

Définition 4.23 ($\lambda!$-calcul). —

- *Les termes sont définis par la grammaire 4.2 où x s'étend sur un ensemble*

$$\begin{array}{rcll} t, u & ::= & x & \textit{variable} \\ & | & \lambda x.t & \textit{abstraction} \\ & | & tu & \textit{application} \\ & | & !t & \textit{modalité bien-sûr} \\ & | & \mathrm{der}\, t & \textit{déréliction} \\ & | & t[x := u] & \textit{substitution explicite} \end{array}$$

Grammaire 4.2 : $\lambda!$-calcul

  dénombrable de variables, pris, par commodité, pour l'ensemble des noms cartésiens.

- *Pour obtenir les contextes, il suffit d'ajouter un trou $\{-\}$ à la grammaire.*

- *La notation* $t[-]$ *désigne un terme de la forme* $t[x_1 := u_1]\cdots[x_n := u_n]$, *où* n *peut être nul.*
- *Le constructeur* $!(-)$ *est prioritaire sur les constructeurs binaires, i.e. ,* $!tu$ *et* $!t[x := u]$ *doivent être compris comme* $(!t)u$ *et* $(!t)[x := u]$, *respectivement.*

Définition 4.24 (Réduction $\lambda!$). — *La réduction est la fermeture contextuelle des règles suivantes :*

$$\begin{aligned}(\lambda x.t)[-]u &\to t[x := u][-]\\ t[x := !u[-]] &\to t\{u/x\}[-]\\ \mathrm{der}(!t[-]) &\to t[-]\end{aligned}$$

où $t\{u/x\}$ *désigne la substitution, sans capture, usuelle de* u *à toutes les occurrences libres de* x *dans* t.
Ceci induit un système de réduction $\Lambda_!$.
Nous définissons inductivement une famille d'applications $(\!|-|\!)a : \Lambda_! \to \mathcal{P}roc$ *paramétrées par un nom linéaire* a *:*

$$\begin{aligned}(\!|x|\!)a &:= \overline{x}\langle a\rangle & (\!|\lambda x.t|\!)a &:= a\langle c,d\rangle \mid \overline{c}(x)(\!|t|\!)d\\ (\!|tu|\!)a &:= (\!|t|\!)b \mid \overline{b}\langle c,a\rangle \mid (\!|u|\!)c & (\!|!t|\!)a &:= a(z)!z(b).(\!|t|\!)b\\ (\!|\mathrm{der}\, t|\!)a &:= \overline{c}(z)\overline{z}\langle a\rangle \mid (\!|t|\!)c & (\!|t[x := u]|\!)a &:= \overline{b}(x)(\!|t|\!)a \mid (\!|u|\!)b.\end{aligned}$$

Remarque 4.25. — *Dans [11] est également introduit la* réduction faible *pour le* $\lambda!$*-calcul, qui ne se réduit pas sous* $!(-)$, *notons* $\Lambda_!^w$ *le système de réduction correspondant. Dans ce système nous nous restreignons aux formes normales qui sont des termes dont les redex et les* clashes *(configurations indésirables définies dans [11]) n'apparaissent que sous un* $!(-)$.

Proposition 4.26. — *Pour tout nom linéaire* a, $(\!|-|\!)a$ *est un plongement. De plus, considéré comme une application* $\Lambda_!^w \to \mathcal{P}roc_0$, *c'est un plongement consistant.*

Remarque 4.27 (Sur $\lambda!$ et *call-by-push-value*). — *Les résultats de section 3.4.3 peuvent être formulés directement dans la syntaxe du* $\lambda!$*-calcul. Les arbres de Böhm sont comme attendues : étant donné un* $\lambda!$*-terme* t, *si la réduction faible pour* t *ne se termine pas, alors* $\mathrm{BT}(t) = \bot$. *Sinon, elle se termine sur un terme*

de la forme $\mathsf{C}\{!u_1, \dots, !u_n\}$ *où* C *est un contexte à trous multiples ne contenant pas de* $!(-)$*, alors,* $\mathrm{BT}(t) = \mathsf{C}\{!\mathrm{BT}(u_1), \dots, !\mathrm{BT}(u_n)\}$*. Remarquons que* $\bot$ *et* $!\bot$ *sont des arbres de Böhm différents.*

Le développement de Taylor de call-by-push-value *a déjà été défini et étudié dans [15]. Cela donne ici une reformulation dans le contexte du* $\lambda!$ *avec des substitutions explicites. Les termes d'approximation de Taylor sont définis dans la grammaire 4.3, où* a *s'étend sur les variables linéaires, i.e. , aucune*

$$\begin{array}{rcl} r, s & ::= & a \\ & | & \lambda\langle\widetilde{a}\rangle.r \\ & | & rs \\ & | & \langle r_1, \dots, r_n\rangle \\ & | & \mathrm{der}\, r \\ & | & r[\langle\widetilde{a}\rangle := s] \end{array}$$

Grammaire 4.3 : $\lambda!$ & CbPV

variable n'apparaît deux fois et chaque variable de $\widetilde{a}$ *dans les lieurs* $\lambda\langle\widetilde{a}\rangle.t$ *ou* $t[\langle\widetilde{a}\rangle := u]$ *doit apparaître libre dans* t*. Les règles de réduction sont les suivantes :*

$$\begin{aligned} (\lambda\langle\widetilde{a}\rangle.r)s &\to r[\langle\widetilde{a}\rangle := s], \\ r[\langle\widetilde{a}\rangle := \langle\widetilde{s}\rangle] &\to r\{\widetilde{s}/\widetilde{a}\}, \\ \mathrm{der}\langle r\rangle &\to r, \end{aligned}$$

avec la condition que, dans la deuxième règle, $|\widetilde{a}| = |\widetilde{s}|$*. La relation d'approximation est définie à l'aide des jugements* $\Xi \vdash r \sqsubset t$ *avec* Ξ *consistant en des déclarations de la forme* $a \sqsubset x$*, sans qu'aucune variable linéaire n'apparaisse deux fois dans* Ξ*. La relation est définie par la figure 4.4.*

Par proposition 4.26 et théorème 3.48, nous savons que les arbres de Böhm et les approximations de Taylor ci-dessus interagissent bien. La proposition 4.26 implique également les résultats de section 4.1.2. Nous ne les détaillerons pas ici, mais ils nous permettent d'obtenir immédiatement le système de types intersection de [11], ainsi que la propriété qu'il caractérise les termes avec des formes normales faiblement sans conflit.

$$\frac{}{a \sqsubset x \vdash a \sqsubset x} \qquad \frac{\Xi, \widetilde{a} \sqsubset x \vdash r \sqsubset t}{\Xi \vdash \lambda\langle\widetilde{a}\rangle.r \sqsubset \lambda x.t}$$

$$\frac{\Xi \vdash r \sqsubset t \qquad \Upsilon \vdash s \sqsubset u}{\Xi, \Upsilon \vdash rs \sqsubset tu} \qquad \frac{\Xi_1 \vdash r_1 \sqsubset t \qquad \cdots \qquad \Xi_n \vdash r_n \sqsubset t}{\Xi_1, \ldots, \Xi_n \vdash \langle\widetilde{r}\rangle \sqsubset !t}$$

$$\frac{\Xi \vdash r \sqsubset t}{\Xi \vdash \mathrm{der}\, r \sqsubset \mathrm{der}\, t} \qquad \frac{\Xi, \widetilde{a} \sqsubset x \vdash r \sqsubset t \qquad \Upsilon \vdash s \sqsubset u}{\Xi, \Upsilon \vdash r[\langle\widetilde{a}\rangle := s] \sqsubset t[x := u]}$$

FIG. 4.4 : Approximations de TAYLOR dans le λ!-calcul.

Les articles [15] et [11] considèrent tous deux les plongements bien connus de l'appel-par-nom et de l'appel-par-valeur du λ-calcul dans call-by-push-value, *et extrapolent à partir de ces plongements des notions appropriées d'approximations de TAYLOR et de systèmes de types intersection pour l'appel-par-nom et l'appel-par-valeur, en récupérant les résultats de [27, 29, 22, 44] et [31, 14, 13, 22]. Dans notre cadre, il s'agit de plongements au sens technique de la définition 3.5. Comme les plongements se composent, nous retrouvons également ces résultats de manière uniforme.*

REMARQUE 4.28. — *En outre, nous récupérons les arbres de BÖHM habituels [7] et les arbres de BÖHM appel par valeur de [44], ainsi que les théorèmes de commutation correspondants [27, 29, 44]. Cependant, les arbres de BÖHM ne sont pas n'importe quels processus, ils ont une structure qui est dictée par la syntaxe du calcul plongé dans* $\mathcal{P}\mathrm{roc}$. *En effet, si* $f : \mathcal{S} \to \mathcal{P}\mathrm{roc}$ *est un plongement et si* s *est un terme du calcul* $\mathcal{S}$, $\mathrm{BT}(s)$ *au sens de la définition 3.38 est un ensemble de processus sans coupures de* $\mathcal{P}\mathrm{roc}$ *obtenus à partir de processus de la forme* $f(s')$ *où* s' *parcourt les réduits de* s, *en remplaçant éventuellement des* **0** *là où* f *insère des boîtes.*

Il est donc possible de « tirer-en-arrière » les processus de $\mathrm{BT}(s)$ *le long de* f, *et le réécrire comme des termes (normaux) de* $\mathcal{S}$ *avec éventuellement des* $\bot$ *quelque part, correspondant aux* **0**. *Si cela est fait dans le cas des plongements du λ-calcul appel-par-nom et appel-par-valeur, alors les arbres de BÖHM classiques sont obtenus.*

Reformulé dans l'autre sens : soit un λ-terme t *et définissons son arbre de BÖHM appel-par-nom ou appel-par-valeur de la manière classique, comme un*

idéal $\mathrm{BT}(t)$ *de* λ*-termes normaux avec éventuellement des* $\bot$*. Ensuite, définissons un codage de ces termes avec* $\bot$ *dans* $\mathcal{P}roc$*, qui est exactement le plongement* f *appel-par-nom ou appel-par-valeur, et qui envoie* $\bot$ *sur le processus* $\mathbf{0}$*. Ainsi, si ce codage est appliqué terme à terme à* $\mathrm{BT}(t)$*, nous obtenons exactement* $\mathrm{BT}(f(t))$ *(de la définition 3.38).*

Cette remarque s'applique à l'identique au développement de Taylor *(définition 3.46). Les approximations de* Taylor *sont des processus de* $\mathcal{P}roc$ *qui approximent (figure 3.1) des traduits de termes du calcul de départ. La structure du calcul source se réfléchit donc dans la structure de ces approximants, et il est en général facile de les décrire directement dans une sytaxe similaire à celle du langage source.*

4.3 Logique classique

Définition 4.29. — *Le* calcul des piles *[12] est un calcul pour le calcul classique, intégrant le* $\lambda\mu$*-calcul. Sa syntaxe comporte* piles π*,* termes t *et* processus P*, et est donnée par la grammaire 4.4, où* α *s'étend sur un ensemble infini*

$$\begin{array}{rcl} \pi & ::= & \alpha \\ & | & t \cdot \pi \\ & | & \mathsf{tl}(\pi) \\ t & ::= & \mu\alpha.P \\ & | & \mathsf{hd}(\pi) \\ P & ::= & \langle t, \pi \rangle \end{array}$$

Grammaire 4.4 : Calcul des piles

de variables de pile*, considéré comme un sous-ensemble des noms cartésiens de processus.*
La construction $\mu\alpha$ *est un lieur. La réduction est définie donnée par :*

$$\langle \mu\alpha.P, \pi \rangle \to P\{\pi/\alpha\} \qquad \mathsf{hd}(t \cdot \pi) \to t \qquad \mathsf{tl}(t \cdot \pi) \to \pi.$$

Définition 4.30 (Plongements pour le calcul des piles). — *Soient* $\mathcal{S}tk$*,* $\mathcal{T}erm$ *et* $\mathcal{S}tk\mathcal{P}roc$ *les systèmes de réduction induits par la grammaire 4.4, avec respectivement les piles, les termes et les processus comme objets. Nous définissons deux*

familles d'applications $(\!|-|\!)a : \mathcal{S}tk \to \mathcal{P}roc$ *et* $(\!|-|\!)a : \mathcal{T}erm \to \mathcal{P}roc$*, paramétrées par un nom linéaire* a*, ainsi qu'une application* $(\!|-|\!) : \mathcal{S}tkProc \to \mathcal{P}roc$ *de la manière suivante :*

$$\begin{aligned}
(\!|\alpha|\!)a &:= \overline{\alpha}\langle a\rangle \\
(\!|t \cdot \pi|\!)a &:= \overline{a}\langle b, c\rangle \mid b(x)!x(d).(\!|t|\!)d \mid c(y)!y(e).(\!|\pi|\!)e \\
(\!|\mu\alpha.P|\!)a &:= \overline{a}(\alpha)(\!|P|\!) \\
(\!|\mathsf{hd}(\pi)|\!)a &:= (\!|\pi|\!)b \mid b\langle c, d\rangle \mid \overline{c}(x)\overline{d}(y)(\overline{x}\langle a\rangle) \\
(\!|\langle t, \pi\rangle|\!) &:= (\!|t|\!)a \mid a(x)!x(b).(\!|\pi|\!)b \\
(\!|\mathsf{tl}(\pi)|\!)a &:= (\!|\pi|\!)b \mid b\langle c, d\rangle \mid \overline{c}(x)\overline{d}(y)(\overline{y}\langle a\rangle).
\end{aligned}$$

Proposition 4.31. — *Les applications de la définition 4.30 précédente sont des plongements.*

Remarque 4.32. — *Comme mentionné dans l'introduction, l'encodage du* $\lambda\mu$*-calcul dans le calcul des piles n'est pas un plongement dans notre sens technique, nous ne pouvons donc pas appliquer directement nos résultats au* $\lambda\mu$*-calcul.*

Néanmoins, nous disposons maintenant d'une nouvelle théorie opérationnelle des arbres de Böhm et du développement de Taylor pour un calcul Curry-Howard-isomorphe à la logique classique (ce qui est mentionné comme question ouverte dans [5] pour le contexte du $\lambda\mu$*-calcul). Nous laissons l'étude de cette théorie, en particulier la signification des arbres de Böhm, à des travaux futurs.*

4.4 Calculs concurrents

Définition 4.33. — *Le* π-calcul polyadique asynchrone *est défini par la grammaire 4.5, où nous supposons que les noms sont les noms cartésiens des processus (de* $\mathcal{P}roc$*).*

La congruence structurelle et la réduction sont standard, avec les règles suivantes :

$$\begin{aligned}
\overline{x}\langle\widetilde{y}\rangle \mid !x(\widetilde{z}).P &\to P\{\widetilde{y}/\widetilde{z}\} \mid !x(\widetilde{z}).P \\
\nu x(!x(\widetilde{y}).P \mid Q) &\to \nu x Q
\end{aligned}$$

$$\begin{array}{rcl} P, Q & ::= & \mathbf{0} \\ & | & P \mid Q \\ & | & \nu x P \\ & | & \overline{x}\langle \widetilde{y} \rangle \\ & | & !x(\widetilde{y}).P \end{array}$$

Grammaire 4.5 : π-calcul polyadique asynchrone.

à condition, dans la deuxième règle, que x ne soit pas le sujet d'une sortie dans Q.

Remarque 4.34. — *Nous considérons ici la variante* hyperlocalisée *du calcul [17], qui est définie en se limitant aux processus tels que, dans $!x(\widetilde{y}).P$, aucun nom libre de P n'apparaît en tant que sujet d'une entrée. En outre, la réduction n'est autorisée que dans le cadre d'une restriction. Comme le montre [17], il s'agit d'un calcul raisonnablement expressif, avec un non-déterminisme complet ainsi que des verrous.*

Soit Π le système de réduction correspondant au calcul ci-dessus. En utilisant la notation $(z \Rightarrow y) := !z(c).\overline{y}\langle c \rangle$ (en tant que processus de $\mathcal{P}roc$), nous définissons une application $(\!|-|\!) : \Pi \to \mathcal{P}roc_0$ en laissant

$$\begin{aligned} (\!|\overline{x}\langle y_1, \ldots, y_n \rangle|\!) &:= \overline{x}\langle a \rangle \mid \overline{a}\langle b_1, \ldots, b_n \rangle \mid b_1(z_1)(z_1 \Rightarrow y_1) \mid \cdots \mid b_n(z_n)(z_n \Rightarrow y_n) \\ (\!|!x(y_1, \ldots, y_n).P|\!) &:= !x(a).(a\langle b_1, \ldots, b_n \rangle \mid \overline{b_1}(y_1) \cdots \overline{b_n}(y_n)(\!|P|\!)) \end{aligned}$$

et en faisant en sorte que $(\!|-|\!)$ agisse de manière homomorphe sur $\mathbf{0}$, la composition parallèle et la restriction.

Proposition 4.35. — *L'application $(\!|-|\!)$ est un plongement.*

Remarque 4.36. — *Le plongement ci-dessus n'est cependant pas consistant au regard de toute notion raisonnable de forme normale pour le π-calcul, car $(\!|P|\!)$ n'est pas nécessairement correct (les processus peuvent avoir toutes sortes de cercles vicieux).*

Cela n'empêche pas de prendre le système de types intersection de section 4.1.2 et de l'utiliser comme point de départ pour trouver un système de types intersection fonctionnel pour le π-calcul hyperlocalisé. C'est exactement la genèse de l'article [17].

Conclusion

Nous avons généralisé dans cette thèse un résultat central en théorie des approximations de programmes en proposant une approche axiomatique des approximations. Pour ce faire, nous avons proposé un cadre pour parler de plongements entre calculs. Nous avons montré que le théorème de commutation entre les approximation de Böhm et de Taylor, dû à Ehrhard et Regnier, est une instance du théorème 3.48 mais « tiré-en-arrière ». Une fois notre théorie posée, nous avons donc pu l'appliquer à différents calculs de la littérature.

Pour ce faire, notre approche a été d'introduire un calcul de processus, $\mathcal{P}roc$, inspiré de la logique linéaire différentielle, puis d'introduire les notions d'approximations de Böhm et de Taylor dans ce calcul, et de démontrer le théorème de commutation entre les deux dans son cas particulier. Utilisé de concert avec la notion de plongement, cela permet de relever ces des outils de la logique linéaire dans d'autres calculs : $\lambda!$-calcul, $\lambda\mu$-calcul, calculs concurrents.

Concernant l'applicabilité de ces résultats, il est à souligner que le fait qu'un plongement $\mathcal{S} \to \mathcal{P}roc$ établisse immédiatement une élégante théorie des approximations de Böhm et de Taylor pour $\mathcal{S}$ ne dit rien sur l'intérêt réel de cette théorie. Tout d'abord, en l'état, la théorie est formulée dans la syntaxe de $\mathcal{P}roc$, et la reformuler dans une syntaxe « adaptée à $\mathcal{S}$ » n'est pas automatique : le cas de la section 4.2 avec *call-by-push-value* en est un exemple. Deuxièmement, sa pertinence et son utilité doivent être vérifiées au cas par cas. Par exemple, nous n'avons pas encore étudié les arbres de Böhm pour le calcul des piles ou pour le fragment du calcul π présenté dans la section 4.3 et la sec-

tion 4.4.

Enfin, une emphase est à mettre concernant la notion de plongement proposée à la définition 3.5. Bien que cette notion suffise pour les exemples présentés ici, de nombreux encodages, en particulier des calculs de processus, ne sont pas des plongements au sens technique décrit. Généralement parce qu'ils sont à équivalence près. Par exemple, l'encodage du $\lambda\mu$-calcul dans le calcul des piles mentionné dans la section 4.3 est à β-équivalence, et n'est donc pas directement couvert. L'extension de ces résultats de la section 3.4.3 et la section 4.1.2 à une classe plus générale de plongements est un sujet intéressant pour les travaux futurs.

Bibliographie

[1] Beniamino Accattoli. « Exponentials as Substitutions and the Cost of Cut Elimination in Linear Logic ». In : *Log. Methods Comput. Sci.* 19.4 (2023).

[2] Beniamino Accattoli, Ugo Dal Lago et Gabriele Vanoni. « The (In)Efficiency of Interaction ». In : *Proc. ACM Program. Lang.* 5.POPL (2021). DOI : 10.1145/3434332.

[3] Beniamino Accattoli, Ugo Dal Lago et Gabriele Vanoni. « The Machinery of Interaction ». In : *Proceedings of the 22nd International Symposium on Principles and Practice of Declarative Programming*. PPDP '20. New York, NY, USA : Association for Computing Machinery, 2020. ISBN : 9781450388214. DOI : 10.1145/3414080.3414108.

[4] Beniamino Accattoli, Ugo Dal Lago et Gabriele Vanoni. *The Space of Interaction (long version)*. 2021. eprint : 2104.13795.

[5] Davide Barbarossa. « Resource approximation for the λ-μ-calculus ». In : *Proceedings of LICS*, 27 :1-27 :12.

[6] Davide Barbarossa et Giulio Manzonetto. « Taylor Subsumes Scott, Berry, Kahn and Plotkin ». In : *PACMPL* 4.POPL (2020), 1 :1-1 :23. DOI : 10.1145/3371069. URL : https://doi.org/10.1145/3371069.

[7] Henk Barendregt. « The Type Free Lambda Calculus ». In : *Handbook of Mathematical Logic*. Sous la dir. de Jon Barwise. T. 90. Stu-

dies in Logic and the Foundations of Mathematics. Elsevier, 1977, p. 1091-1132.

[8] Henk Barendregt, Mario Coppo et Mariangiola Dezani-Ciancaglini. « A Filter Lambda Model and the Completeness of Type Assignment ». In : *J. Symb. Log.* 48.4 (1983), p. 931-940.

[9] Henkrik Pieter Barendregt. *The Lambda Calculus : Its Syntax and Semantics*. Revised. T. 103. Studies in Logic and The Foundations of Mathematics. Elsevier Science, 1984.

[10] Gérard Boudol. « The Lambda-Calculus with Multiplicities ». In : *Proceedings of CONCUR*. T. 715. Lecture Notes in Computer Science. Springer, 1993, p. 1-6.

[11] Antonio Bucciarelli et al. « The bang calculus revisited ». In : *Inf. Comput.* 293 (2023), p. 105047.

[12] Alberto Carraro, Thomas Ehrhard et Antonino Salibra. « The Stack calculus ». In : *Proceedings of LSFA*. T. 113. EPTCS. 2012, p. 93-108.

[13] Daniel de Carvalho. « Execution time of λ-terms via denotational semantics and intersection types ». In : *Math. Struct. Comput. Sci.* 28.7 (2018), p. 1169-1203.

[14] Daniel de Carvalho. « Sémantiques de la logique linéaire et temps de calcul ». Ph.D. thesis. Université de la Méditerranée–Aix-Marseille 2, 2007.

[15] Jules Chouquet et Christine Tasson. « Taylor expansion for Call-By-Push-Value ». In : *Proceedings of CSL*. T. 152. LIPIcs. 2020, 16 :1-16 :16.

[16] Mario Coppo et Mariangiola Dezani-Ciancaglini. « An extension of the basic functionality theory for the λ-calculus ». In : *Notre Dame J. Formal Log.* 21.4 (1980), p. 685-693.

[17] Ugo Dal Lago et al. « Intersection Types and Runtime Errors in the Pi-Calculus ». In : *Proceedings of the ACM on Programming Languages* 3.POPL :7 (2019).

[18] Vincent Danos et Laurent Regnier. « The structure of multiplicatives ». In : *Arch. Math. Log.* 28.3 (1989), p. 181-203.

[19] Aloÿs Dufour et Damiano Mazza. « Böhm and Taylor for All ! » In : *Proceedings of FSCD*. 2024, 26 :1-26 :20.

[20] Thomas Ehrhard. « An introduction to differential linear logic : proof-nets, models and antiderivatives ». In : *Math. Struct. Comput. Sci.* 28.7 (2018), p. 995-1060.

[21] Thomas Ehrhard. « Call-By-Push-Value from a Linear Logic Point of View ». In : *Proceedings of ESOP*. T. 9632. Lecture Notes in Computer Science. 2016, p. 202-228.

[22] Thomas Ehrhard. « Collapsing non-idempotent intersection types ». In : *Proceedings of CSL*. T. 16. LIPIcs. 2012, p. 259-273.

[23] Thomas Ehrhard. « Finiteness spaces ». In : *Math. Struct. Comput. Sci.* 15.4 (2005), p. 615-646.

[24] Thomas Ehrhard. « On Köthe Sequence Spaces and Linear Logic ». In : *Math. Struct. Comput. Sci.* 12.5 (2002), p. 579-623.

[25] Thomas Ehrhard et Giulio Guerrieri. « The Bang Calculus : an untyped lambda-calculus generalizing call-by-name and call-by-value ». In : *Proceedings of PPDP*. 2016, p. 174-187.

[26] Thomas Ehrhard et Olivier Laurent. « Acyclic Solos and Differential Interaction Nets ». In : *Log. Methods Comput. Sci.* 6.3 (2010).

[27] Thomas Ehrhard et Laurent Regnier. « Böhm trees, Krivine's machine and the Taylor expansion of λ-terms ». In : *Proceedings of Conference on Computability in Europe (CiE)*. T. 3988. Lecture Notes in Computer Science. Springer, 2006, p. 186-197.

[28] Thomas Ehrhard et Laurent Regnier. « Differential interaction nets ». In : *Theor. Comput. Sci.* 364.2 (2006), p. 166-195.

[29] Thomas Ehrhard et Laurent Regnier. « Uniformity and the Taylor expansion of ordinary lambda-terms ». In : *Theor. Comput. Sci.* 403.2-3 (2008), p. 347-372.

[30] Thomas Ehrhard, Christine Tasson et Michele Pagani. « Probabilistic Coherence Spaces Are Fully Abstract for Probabilistic PCF ». In : *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '14. New York, NY, USA : Association for Computing Machinery, 2014, p. 309–320. ISBN : 9781450325448. DOI : 10.1145/2535838.2535865.

[31] Philippa Gardner. « Discovering Needed Reductions Using Type Theory ». In : *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19-22, 1994, Proceedings*. Sous la dir. de Masami Hagiya et John C. Mitchell. T. 789. Lecture Notes in Computer Science. 1994, p. 555-574. DOI : 10.1007/3-540-57887-0_115.

[32] Jean-Yves Girard. « Linear Logic ». In : *Theor. Comput. Sci.* 50 (1987), p. 1-102.

[33] Jean-Yves Girard, Yves Lafont et Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[34] Kohei Honda et Olivier Laurent. « An exact correspondence between a typed pi-calculus and polarised proof-nets ». In : *Theor. Comput. Sci.* 411.22-24 (2010), p. 2223-2238.

[35] Zhengfeng Ji et al. *MIP*=RE*. 2022. arXiv : 2001.04383 [quant-ph]. URL : https://arxiv.org/abs/2001.04383.

[36] Yves Lafont. « Linear Logic Pages ». 1999.

[37] Paul Blain Levy. *Call-By-Push-Value : A Functional/Imperative Synthesis*. T. 2. Semantics Structures in Computation. Springer, 2004.

[38] Damiano Mazza. *An axiomatic notion of approximation for programming languages and machines*. 2021. eprint : 2104.13795. URL : https://lipn.univ-paris13.fr/~mazza/papers/ApxAxiom.pdf.

[39] Damiano Mazza. « Polyadic Approximations in Logic and Computation ». *Habilitation* thesis. Université Paris 13, 2017.

[40] Damiano Mazza. « The true concurrency of differential interaction nets ». In : *Math. Struct. Comput. Sci.* 28.7 (2018), p. 1097-1125.

[41] Damiano Mazza et Luc Pellissier. « A Functorial Bridge between the Infinitary Affine λ-Calculus and Linear Logic ». In : *Proceedings of ICTAC*. 2015, p. 140-161.

[42] Damiano Mazza, Luc Pellissier et Pierre Vial. « Polyadic Approximations, Fibrations and Intersection Types ». In : *Proceedings of the ACM on Programming Languages* 2.POPL :6 (2018).

[43] Paul-André Melliès. « Categorical semantics of linear logic ». In : *Interactive models of computation and program behaviour*. Panoramas et Synthèses 27. Société Mathématique de France, 2009, p. 1-196.

[44] Michele Pagani, Giulio Manzonetto et Axel Kerinec. « Revisiting Call-by-value Böhm trees in light of their Taylor expansion ». In : *Logical Methods in Computer Science* 16 (2020).

[45] Michele Pagani et Paolo Tranquilli. « The conservation theorem for differential nets ». In : *Math. Struct. Comput. Sci.* 27.6 (2017), p. 939-992.

[46] Davide Sangiorgi. *Introduction to bisimulation and coinduction*. Cambridge University Press, 2011.

[47] Terese. *Term Rewriting Systems*. T. 55. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

[48] Paolo Tranquilli. « Confluence of Pure Differential Nets with Promotion ». In : *Proceedings of CSL*. T. 5771. Lecture Notes in Computer Science. 2009, p. 500-514.

[49] Takeshi Tsukada, Kazuyuki Asada et C.-H. Luke Ong. « Generalised species of rigid resource terms ». In : *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2017, p. 1-12. DOI : 10.1109/LICS.2017.8005093.