Investigations classiques, complexes et concurrentes à l'aide de la logique linéaire

Olivier Laurent

HABILITATION À DIRIGER DES RECHERCHES

Université Paris Diderot – Paris 7

Investigations classiques, complexes et concurrentes à l'aide de la logique linéaire

Olivier LAURENT

5 février 2010

Jury :	Pierre-Louis	Curien	(rapporteur)
	Jean-Yves	Girard	
	Jean	Goubault-Larrecq	(président)
	Gordon	Plotkin	
	Peter	Selinger	(rapporteur)
	Glynn	Winskel	(rapporteur)

Table des matières

Remerciements

- Un grand merci à Ugo Dal Lago, Thomas Ehrhard, Russ Harmer, Kohei Honda et Lorenzo Tortora de Falco qui furent mes co-auteurs pour les travaux présentés ici : ce travail est aussi le leur. Merci également à tous mes autres collaborateurs.
- Merci aux étudiants qui ont accepté de travailler avec moi (Jérôme Rocheteau, Joachim de Lataillade, Brian Hill, Marc Lasson et Pierre-Marie Pédrot).
- Merci aux rapporteurs (Pierre-Louis Curien, Peter Selinger et Glynn Winskel) et aux autres membres du jury (Jean-Yves Girard, Jean Goubault-Larrecq et Gordon Plotkin) qui ont eu la gentillesse d'accepter de me consacrer un peu de leur temps précieux.
- Merci au laboratoire PPS et à tous ses membres : ce fut un grand plaisir d'y travailler. Merci également aux autres chercheurs avec lesquels j'ai pu partager mes intérêts scientifiques (en France, je pense évidemment au LIP mais également à l'IML, au LAMA, au LIPN et au LIX).
- Merci également à tous ceux qui lisent ces remerciements dans l'espoir d'y trouver leur nom.

La logique linéaire n'a déjà plus 20 ans ! Ce système logique introduit par Jean-Yves Girard [Gir87] a fait profondément évoluer les rapports entre logique et calcul autour de la correspondance de Curry-Howard. Issue de la découverte d'un lien exact entre les preuves en déduction naturelle intuitionniste minimale et les λ-termes simplement typés, cette correspondance donne une signification calculatoire aux preuves formelles et une justification logique aux primitives de programmation. L'évaluation des programmes coïncide ainsi avec la procédure d'élimination des coupures du calcul des séquents (ou la normalisation en déduction naturelle). Il s'agit d'une transformation des preuves qui permet d'extraire l'« essence » d'une preuve formelle comme l'exprime la propriété de la sous-formule : une preuve sans coupure n'utilise que des sous-formules de la formule prouvée. On en déduit immédiatement la cohérence (non contradiction) du système logique concerné, ce qui était la motivation des logiciens pour définir l'élimination des coupures.

La *logique linéaire* (LL) a été introduite grâce à l'analyse des modèles dénotationnels de la logique intuitionniste (principalement les espaces cohérents) et amène une décomposition de l'implication intuitionniste via la traduction de Girard : $A \to B = !A \multimap B$ $(= ?A^{\perp} \parr B)$. L'effet principal de cette décomposition est de localiser l'utilisation des règles structurelles (contraction = copie, affaiblissement = effacement) au niveau des connecteurs *exponentiels* ! et ?. L'impossibilité d'utiliser les règles structurelles aussi librement qu'en logique classique a pour effet de dupliquer les connecteurs : deux conjonctions $\otimes$ et &, deux disjonctions $\parr$ et $\oplus$, ... On obtient ainsi la famille des connecteurs *multiplicatifs* $\otimes$, $\parr$, 1, $\perp$ (on peut également ajouter $\multimap$ défini par $A \multimap B = A^{\perp} \parr B$) et celle des connecteurs *additifs* &, $\oplus$, $\top$, 0.

Nous nous intéressons à l'étude de la correspondance de Curry-Howard par le biais de la logique linéaire ainsi qu'aux applications en logique et en informatique. Nous utiliserons pour cela les trois principales « variantes » de la logique linéaire :

- La *logique linéaire polarisée* est le principal système issu de la logique linéaire pour analyser le contenu calculatoire de la logique classique. Avec d'autres systèmes équivalents (comme le $\lambda\mu$-calcul notamment), cette logique permet d'amener la correspondance de Curry-Howard pour la logique classique à la maturité qu'on connaît dans le cadre intuitionniste : modèles catégoriques, sémantique des jeux, ...
- Les *logiques linéaires allégées* permettent de prendre en compte en logique des contraintes de complexité sur l'évaluation des programmes. Ce point est évidemment pertinent lorsqu'il s'agit d'exécuter des programmes possiblement coûteux en temps et en espace par rapport aux ressources disponibles.
- La *logique linéaire différentielle* amène à l'extension de la correspondance de Curry-Howard à la programmation concurrente et distribuée à travers une interprétation logique des calculs de processus.

Il s'agit de trois lignes de travaux assez indépendantes dans la mesure où, bien que proches, ces systèmes dérivés de la logique linéaire ne sont, à l'heure actuelle, pas intégralement rendus compatibles. Néanmoins, les outils sous-jacents restent bien souvent les mêmes : réseaux de preuve, modèle relationnel, sémantique des jeux, ...

La thèse défendue ici est que la logique linéaire (notamment via les trois variantes mentionnées ci-dessus) apparaît comme un outil central dans l'étude de la correspondance de Curry-Howard et est en mesure de fournir un cadre unifié. En effet, l'unification entre approches (logique, informatique, syntaxique, sémantique, ...), l'épuration des concepts, la découverte de structures simples et élégantes sont le cœur de notre démarche.

Première partie

Logiques classiques constructives[1]

[Lau05a] Olivier Laurent. Classical isomorphisms of types. *Mathematical Structures in Computer Science*, 15(5):969–1004, October 2005.

[HL06] Russ Harmer and Olivier Laurent. The anatomy of innocence revisited. In S. Arun-Kumar and Naveen Garg, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 224–235. Springer, December 2006.

[Lau08] Olivier Laurent. Game semantics for first-order logic. Submitted to *Logical Methods in Computer Science*, November 2008.

[1] Cette partie aurait pu s'intituler « Logique linéaire polarisée et logique classique » mais, forts des relations désormais connues entre les différents systèmes de logique classique constructive, nous utiliserons plutôt le $\lambda\mu$-calcul comme langage (la logique linéaire polarisée aurait joué un rôle parfaitement similaire [Lau05b]).

La correspondance de Curry-Howard permet une unification entre l'étude des systèmes logiques en théorie de la démonstration et l'étude des langages de programmation. Initialement restreinte au cas de la logique intuitionniste, du λ-calcul et des langages purement fonctionnels, de nombreuses extensions sont désormais à notre disposition. Celle qui nous intéresse ici concerne la logique classique et les opérateurs de contrôle (comme `call/cc`). Les progrès faits dans l'étude syntaxique de ces systèmes et dans celle de leur sémantique dénotationnelle (ce qui amène naturellement à modifier, raffiner, enrichir, ... les systèmes eux-mêmes) offrent une seconde unification [Gir99] : les formes canoniques syntaxiques deviennent très proches des structures manipulées par les modèles.

Après avoir décrit les éléments techniques qui sous-tendent cette approche, nous présentons le cas concret du rapport entre la logique classique constructive du premier ordre (représentée par une extension du $\lambda\mu$-calcul [Par92]) et un modèle de jeux à la Hyland-Ong/Nickau [HO00].

1 Syntaxe vs. sémantique

On relie traditionnellement systèmes syntaxiques et modèles par les théorèmes de *correction* (« ce qui est prouvable est vrai ») et de *complétude* (« ce qui est vrai est prouvable »). Une évolution due à la correspondance de Curry-Howard, aux travaux développés en sémantique dénotationnelle, ... amène à raffiner ces théorèmes. Considérons un système logique L définissant des formules A et des preuves π de ces formules (avec la relation $\pi \vdash A$: « π est une preuve de A »), un modèle est alors donné par des structures $\mathcal{A}$ pour les formules (objets si on utilise un cadre catégorique), des structures f pour les preuves (morphismes dans le cadre catégorique), une interprétation $[\![A]\!]$ des formules et $[\![\pi]\!]$ des preuves, et une relation $f \vDash \mathcal{A}$ (« f habite $\mathcal{A}$ »). Dans le contexte traditionnel, les preuves comptent peu et on note $\vdash A$ s'il existe une preuve de A et $\vDash \mathcal{A}$ si $\mathcal{A}$ est habité (souvent sans même faire apparaître de notion d'habitants comme dans les algèbre de Boole où $\vDash \mathcal{A}$ est simplement $\mathcal{A} = 1$). La correction d'un modèle $\mathcal{M}$ pour L est alors $\vdash A \Longrightarrow \vDash [\![A]\!]$ et la complétude (prise plutôt au sens du théorème d'incomplétude de Gödel puisqu'on fait référence ici à un modèle donné et pas globalement à la classe de tous les modèles) est la réciproque $\vDash [\![A]\!] \Longrightarrow \vdash A$. Un premier raffinement consiste à prendre en compte les preuves, et on obtient un énoncé de correction modifié $\pi \vdash A \Longrightarrow [\![\pi]\!] \vDash [\![A]\!]$ et une réciproque appelée *complétude forte* (ou surjectivité) $f \vDash [\![A]\!] \Longrightarrow \exists \pi([\![\pi]\!] = f \wedge \pi \vdash A)$. Une deuxième étape prend en compte une notion d'égalité entre preuves $\pi_1 \cong \pi_2$ (c'est un point crucial dans l'interprétation calculatoire des preuves : si les preuves sont des programmes, elles ne peuvent pas être toutes égales) et raffine la correction : $\pi_1 \cong \pi_2 \vdash A \Longrightarrow [\![\pi_1]\!] = [\![\pi_2]\!] \vDash [\![A]\!]$ puis mène à l'*injectivité* $[\![\pi_1]\!] = [\![\pi_2]\!] \vDash [\![A]\!] \Longrightarrow \pi_1 \cong \pi_2 \vdash A$ comme réciproque.

Une opération fondamentale sur les preuves est la *composition* donnée par l'élimination d'une coupure introduite entre une preuve de $A \vdash B$ et une preuve de $B \vdash C$ (on suppose désormais la prouvabilité dans la logique L étendue à des séquents). Ainsi, dans les cas qui nous intéressent, les modèles $\mathcal{M}$ viennent naturellement avec une structure catégorique et toutes les propriétés que l'on vient de voir se reformulent de manière particulièrement adaptée si on s'intéresse, côté syntaxique, à la catégorie $\mathcal{S}$ dont les objets sont les formules et dont les morphismes de A dans B sont les classes d'équivalence de preuves de $A \vdash B$ (idéalement des représentants canoniques de ces classes d'équivalence). La forme la plus forte de correction présentée ci-dessus est la fonctorialité de l'interprétation de $\mathcal{S}$ dans $\mathcal{M}$. La complétude forte requiert que le foncteur soit plein, et l'injectivité que le foncteur soit fidèle. Si, de surcroît, on demande à l'interprétation des formules d'être essentiellement surjective (*i.e.* que, pour tout objet $\mathcal{A}$ de $\mathcal{M}$, il existe une formule A de $\mathcal{S}$ telle que $\mathcal{A} \simeq [\![A]\!]$ où $\simeq$ est l'isomorphisme catégorique dans $\mathcal{M}$), alors on obtient exactement que les catégories $\mathcal{S}$ et $\mathcal{M}$ sont des catégories dites *équivalentes*. Arrivés à une telle correspondance, la distinction entre syntaxe et sémantique devient extrêmement ténue

et permet une unification des méthodes logiques.

Revenons sur la notion d'égalité entre preuves. Les égalités du λ-calcul (la β-égalité et la η-égalité) s'orientent naturellement, dans le cas typé, en β-réduction et η-expansion permettant de définir un représentant canonique de chaque classe de $\beta\eta$-équivalence : la forme β-normale η-longue. À travers Curry-Howard, ces transformations correspondent à l'élimination des coupures et à l'expansion des axiomes en calcul des séquents. Il est cependant clair que les formes normales que l'on obtient avec ces deux ré-écritures ne sont pas canoniques en calcul des séquents : il existe des formes normales qui ne diffèrent que par des transformations insignifiantes (et sont identifiées par tout modèle raisonnable). La propriété d'injectivité est ainsi principalement une question sur la syntaxe (on doit être en mesure de représenter les preuves de manière canonique) alors que la surjectivité nécessite de travailler sur le modèle pour trouver des caractérisations internes des morphismes issus de preuves. Obtenir l'équivalence catégorique mentionnée plus haut nécessite ainsi de travailler à la fois côté syntaxique pour obtenir une « bonne syntaxe » et côté sémantique pour obtenir un « bon modèle ».

On a parlé d'égalité de preuves, une autre question concerne l'égalité entre formules dans une logique donnée. La réponse la plus simple est l'équiprouvabilité : A et B sont égales s'il existe une preuve de $A \vdash B$ et une preuve de $B \vdash A$. Cependant cette notion est beaucoup trop violente, ne reflète aucun contenu calculatoire et identifie par exemple toute formule prouvable à la formule « vrai ». Si l'on considère ces deux preuves de $A \vdash B$ et de $B \vdash A$ comme des conversions des « habitants » de A (ou des preuves de A) en « habitants » de B, respecter l'information calculatoire impose qu'un aller/retour de A à A en passant par B (c'est-à-dire la preuve de $A \vdash A$ obtenue par coupure à partir de $A \vdash B$ et de $B \vdash A$) la préserve (et de même de B à B via A). Au sein de la catégorie syntaxique $\mathcal{S}$ décrite plus haut, cela revient à demander que les deux preuves de $A \vdash A$ et $B \vdash B$ obtenues par composition soient l'identité sur A et sur B. Autrement dit qu'il s'agisse d'isomorphismes entre A et B et donc que A et B soient isomorphes dans $\mathcal{S}$. On parle souvent d'*isomorphismes de types*. L'objectif est alors de caractériser (par exemple par un système équationnel) cette égalité. Cette question, naturelle dès qu'on a muni la syntaxe d'une structure catégorique, a des applications informatiques importantes pour comprendre quand un composant logiciel de type A peut être utilisé dans un contexte où on recherche un composant de type B. Par exemple, si $A \simeq B$, il est facile de construire un programme effectuant une conversion de A en B et d'utiliser ainsi du code existant de type A avec le type B sans modifier son comportement calculatoire. Des caractérisations des isomorphismes de types ont été données pour différents systèmes intuitionnistes [DC95] par des méthodes syntaxiques ou sémantiques. La méthode sémantique consiste à construire un modèle du système concerné (ce qui entraîne automatiquement que tout isomorphisme syntaxique est un isomorphisme dans le modèle, mais il y a généralement plus d'isomorphismes dans le modèle que dans la syntaxe) dont les isomorphismes soient caractérisables et pas plus nombreux que dans la syntaxe. Le fait qu'un modèle n'ajoute pas d'isomorphismes de types par rapport à la syntaxe est une propriété d'injectivité au niveau des formules. En particulier un modèle qui est catégoriquement équivalent à la syntaxe n'introduit jamais d'isomorphismes additionnels.

2 Logique linéaire polarisée, $\lambda\mu$-calcul, ...

Importante propriété découverte par J.-M. Andreoli [And90] quelques années après l'introduction de la logique linéaire, la *focalisation* induit une partition des connecteurs linéaires en deux familles : les *positifs* ($\otimes$, $\oplus$, 0, 1, $\exists$) et les *négatifs* ($\parr$, &, $\bot$, $\top$, $\forall$). On note alors que les isomorphismes de la logique linéaire (associativités, distributivités, éléments neutres, ...) vivent à l'intérieur de ces classes de connecteurs sans en sortir ($A \otimes (B \oplus C) \simeq (A \otimes B) \oplus (A \otimes C)$

par exemple). Seule exception notable, l'isomorphisme $!(A \,\&\, B) \simeq !A \otimes !B$ fait des connecteurs exponentiels une passerelle entre les deux mondes (ici ! va de négatif à positif). De la focalisation (c'est-à-dire, au sens strict, la possibilité de considérer, au sein de LL, un groupe de connecteurs positifs comme un seul connecteur), on passe à la *polarisation* en imposant que les utilisations des connecteurs ne passent pas d'une classe à l'autre sauf via les connecteurs exponentiels. Formellement on restreint les formules de logique linéaire à la grammaire suivante :

$$\begin{array}{lllllllllllll} N & ::= & X & | & \bot & | & \top & | & N \parr N & | & N \,\&\, N & | & \forall x N & | & \forall X N & | & ?P \\ P & ::= & X^\bot & | & 1 & | & 0 & | & P \otimes P & | & P \oplus P & | & \exists x P & | & \exists X P & | & !N \end{array}$$

Utilisant la focalisation et la polarisation, J.-Y. Girard a montré que les connecteurs négatifs préservent les structures de $\parr$-monoïdes ($A \parr A \to A$ et $\bot \to A$), et dualement les positifs préservent les $\otimes$-comonoïdes ($A \to A \otimes A$ et $A \to 1$), et a ainsi construit le système LC [Gir91]. Grâce à ce travail, puis à l'introduction du $\lambda\mu$-calcul par M. Parigot [Par92], a débuté la théorie de la *logique classique constructive* : la logique classique est compatible avec une élimination déterministe (*i.e.* confluente) des coupures lui donnant un sens calculatoire correspondant aux opérateurs de contrôle comme montré par T. Griffin [Gri90].

Après une multiplication des systèmes de logique classique constructive à la fin des années 90, de nombreuses équivalences entre eux ont été mises en évidence (voir par exemple [LQTdF05, Roc05]). Si on se concentre sur les systèmes dits « en appel par nom » (auxquels se ramènent ceux « en appel par valeur » à travers la dualité catégorique explicitée par P. Selinger [Sel01]), on constate qu'ils sont équivalents et que, en ce qui nous concerne en particulier ici où nous nous focalisons sur la syntaxe vue comme catégorie, étudier le $\lambda\mu$-calcul, la logique linéaire polarisée, LC, ... ne fait pas de différence profonde. Si l'on s'intéressait à des études syntaxiques plus fines, et notamment à la réécriture, des différences plus importantes entre systèmes apparaîtraient.

3 Sémantique des jeux

Les modèles de jeux [AJM00, HO00] ont la qualité et le défaut d'être proches de la syntaxe. C'est un défaut dans la mesure où le niveau d'abstraction obtenu est moindre qu'avec d'autres approches. Mais ce sera pour nous une qualité, et c'est ce qui a fait leur succès via les résultats de « full abstraction », car ils fournissent ainsi un outil puissant pour analyser la syntaxe. On s'intéresse ici aux modèles à la Hyland-Ong/Nickau (HO/N) [HO00, Nic94] qui sont particulièrement adaptés aux extensions du λ-calcul typé.

3.1 Réponses

Dans le langage PCF [Plo77], les types sont construits avec le connecteur « flèche » à partir de types de base, en général $\mathbb{N}$ (représentant les entiers) et $\mathbb{B}$ (représentant les booléens). On va autoriser ici de manière générale des types énumérés finis comme types de base, *i.e.* dont les éléments sont donnés par un ensemble fini. Par opposition aux types récursifs par exemple, ces types de base n'ont aucune structure intrinsèque et on se restreint, sans perte de généralité, à un type de base $\mathbb{F}_k$ à k éléments pour chaque entier k. Considérer également des cardinaux infinis et un $\mathbb{F}_\omega$ correspondant à $\mathbb{N}$ ne poserait pas de problème technique mais induirait une syntaxe infinitaire (« en largeur ») que nous préférons éviter ici (de plus ce cas est traité explicitement dans [HL06]).

On considère ainsi le calcul μPCF (une variante de celui décrit par Jim Laird [Lai98] avec notamment, dans l'esprit des arbres de Böhm, des termes potentiellement infinis « en profondeur »

et une construction Ω plutôt qu'un opérateur de point fixe) étendant PCF avec des opérateurs de contrôle à la manière du $\lambda\mu$-calcul.

Les types sont donnés par la grammaire :

$$A ::= \mathbb{F}_k \mid A \to A \qquad (k \in \mathbb{N})$$

et les termes (possiblement infinis en profondeur) par :

$$M ::= a \mid \lambda a.M \mid (M)M \mid \Omega \mid \mathtt{n}_k \mid \mathtt{case}_k\ M\ \mathtt{with}\ \mathtt{0} \mapsto M\ ;\ \ldots\ ;\ \mathtt{k\text{-}1} \mapsto M \mid \mu\alpha.M \mid [\alpha]M$$

où $0 \leq n < k$, avec comme règles de typage :

$$\frac{}{\Gamma, a : A \vdash a : A \mid \Delta} \qquad \frac{\Gamma, a : A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda a.M : B \mid \Delta} \qquad \frac{\Gamma \vdash M : A \to B \mid \Delta \qquad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash (M)N : B \mid \Delta}$$

$$\frac{}{\Gamma \vdash \Omega : A \mid \Delta}$$

$$\frac{}{\Gamma \vdash \mathtt{n}_k : \mathbb{F}_k \mid \Delta}\ n < k \qquad \frac{\Gamma \vdash M : \mathbb{F}_k \mid \Delta \qquad \cdots \quad \Gamma \vdash N_i : A \mid \Delta \quad \cdots \qquad 0 \leq i < k}{\Gamma \vdash \mathtt{case}_k\ M\ \mathtt{with}\ \mathtt{0} \mapsto N_0\ ;\ \ldots\ ;\ \mathtt{k\text{-}1} \mapsto N_{k-1} : A \mid \Delta}$$

$$\frac{\Gamma \vdash M : A \mid \Delta, \alpha : A}{\Gamma \vdash [\alpha]M : \mathbb{F}_0 \mid \Delta, \alpha : A} \qquad \frac{\Gamma \vdash M : \mathbb{F}_0 \mid \Delta, \alpha : A}{\Gamma \vdash \mu\alpha.M : A \mid \Delta}$$

On ne décrit pas ici la théorie équationnelle de ce langage dans la mesure où ce n'est pas le point clef de la discussion que l'on souhaite mener. Cette théorie est décrite dans [Lai98].

Le modèle de *jeux innocents* HO/N de ce langage interprète les types par des *arènes* et les termes typés par des *stratégies*.

Définition (Arène)
Une *arène* est un arbre de *coups*. La racine est une *question* d'*Opposant*, puis les polarités *Joueur*/*Opposant* alternent le long d'une branche de l'arbre. Enfin certaines feuilles de l'arbre (différentes de la racine) sont des *réponses*, les autres coups étant des *questions*.

Pour alléger les notations, on utilisera A pour l'arène interprétant le type A. Le type $\mathbb{F}_k$ est interprété par l'arène ayant une racine $\mathtt{q}$ avec k fils $\mathtt{n}_0$, ..., $\mathtt{n}_{k-1}$ qui sont tous des réponses. Le type $A \to B$ est obtenu à partir de l'arbre interprétant B en greffant l'arbre interprétant A sous la racine de B (ainsi la racine de A devient un fils de la racine de B).

Définition (Vues et Stratégies)
Une *vue* sur l'arène A est une suite de coups de A de polarités alternées commençant par un coup Opposant, telle que tout coup Opposant est un fils du coup Joueur précédent, et munie pour chaque coup Joueur $\mathtt{m}$ d'un pointeur (appelé λ-*pointeur*) vers une occurrence de coup précédente $\mathtt{n}$ où $\mathtt{n}$ est le père de $\mathtt{m}$ dans A.

Une *stratégie* σ est un ensemble non vide de vues de longueurs paires (clos par préfixe pair) tel que si $\mathtt{sm}$ et $\mathtt{sn}$ sont des éléments de σ alors $\mathtt{sm} = \mathtt{sn}$.

En définissant une composition des stratégies et en utilisant les stratégies dites *copycat* comme identités, on obtient une catégorie de contrôle (notion de modèle catégorique de la logique classique constructive introduite dans [Sel01]).

L'interprétation des termes peut être décrite à partir de leur forme β-normale η-longue présentée sous forme d'arbre de Böhm en associant des λ-pointeurs aux liaisons de variables (une occurrence liée λ-pointe vers le λ qui la lie). On obtient ainsi un modèle complet (et injectif) de μPCF.

Théorème (Complétude)
Toute stratégie sur l'arène A est l'interprétation d'un terme clos de μPCF de type A.

Plusieurs travaux [Lai98, DH01, HL06] ont permis d'analyser ce modèle de jeux (en particulier le rôle des réponses) en découpant des sous-catégories qui caractérisent exactement certains sous-langages de μPCF.

Les réponses étant nécessairement des feuilles dans une arène, la définition de vue impose que si une vue contient une réponse Joueur, il s'agit de son dernier coup.

Bon parenthésage. Une vue est *bien parenthésée* si son éventuelle réponse Joueur pointe sur la dernière question Opposant qui la précède. Une stratégie est bien parenthésée si toutes ses vues le sont.

Cette condition caractérise les termes qui n'utilisent pas les constructions $\mu\alpha._-$ et $[\alpha]_-$ puisque l'interprétation d'un tel terme est bien parenthésée et réciproquement toute stratégie bien parenthésée est l'interprétation d'un terme sans $\mu\alpha._-$ ni $[\alpha]_-$ [Lai98].

Rigidité avant. Une vue est *rigide avant* si une réponse Opposant n'est jamais suivie par une question Joueur. Une stratégie est rigide avant si toutes ses vues le sont.

Cette condition caractérise les termes qui n'utilisent pas la construction $\texttt{case}_k$ _ `with` _ puisque l'interprétation d'un tel terme est rigide avant et réciproquement toute stratégie rigide avant est l'interprétation d'un terme sans $\texttt{case}_k$ _ `with` _ (à condition de se donner toutes les fonctions strictes de $\mathbb{F}_k$ dans $\mathbb{F}_p$) [DH01].

Rigidité arrière. Une vue est *rigide arrière* si une réponse Joueur est toujours précédée par une réponse Opposant. Une stratégie est rigide arrière si toutes ses vues le sont.

Cette condition caractérise les termes qui n'utilisent pas la construction $\mathtt{n}_k$ puisque l'interprétation d'un tel terme est rigide arrière et réciproquement toute stratégie rigide arrière est l'interprétation d'un terme sans $\mathtt{n}_k$ (à condition de se donner toutes les fonctions strictes de $\mathbb{F}_k$ dans $\mathbb{F}_p$) [HL06].

Une stratégie à la fois rigide avant et rigide arrière est appelée *rigide*. On peut noter que les stratégies définissant les fonctions strictes de $\mathbb{F}_k$ dans $\mathbb{F}_p$ sont rigides et définissables à l'aide de $\texttt{case}_k$ _ `with` _ et des $\mathtt{n}_p$.

On montre que ces trois conditions sont indépendantes par des *théorèmes de factorisation* :

Théorème (Factorisation du parenthésage)
*Il existe une arène B et une stratégie $\rho_{\mathcal{P}}$ telle que pour toute stratégie σ sur une arène A, il existe une factorisation $\sigma_{\mathcal{P}}$ (sur l'arène $B \to A$) de σ par $\rho_{\mathcal{P}}$ (*i.e. *$\rho_{\mathcal{P}}$ composée avec $\sigma_{\mathcal{P}}$ est égale à σ) qui est bien parenthésée.*

De plus, si σ est rigide avant alors $\sigma_{\mathcal{P}}$ est rigide avant et si σ est rigide arrière alors $\sigma_{\mathcal{P}}$ est rigide arrière.

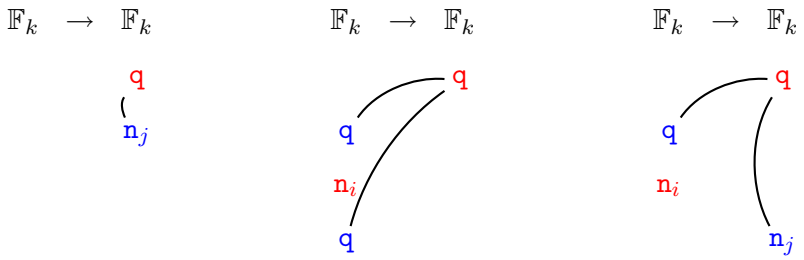
On obtient les théorèmes correspondants pour la rigidité avant et la rigidité arrière. Ces résultats sont traités dans [HL06] dans le cas plus général où l'on utilise le type $\mathbb{F}_\omega$.

Si l'on s'intéresse aux modèles de jeux pour des systèmes logiques, les stratégies obtenues vérifient une contrainte de *totalité* (pendant sémantique de la normalisation) : si $\mathtt{s}$ est une vue de σ stratégie totale sur l'arène A et si $\mathtt{sm}$ est une vue sur A, alors il existe une vue $\mathtt{smn}$ dans σ (qui prolonge $\mathtt{sm}$). Un théorème de complétude montre que les stratégies totales sont l'interprétation des termes de μPCF sans Ω. Un théorème de factorisation montre que toute stratégie σ sur A s'écrit $\sigma = \{\varepsilon\} ; \sigma_{\mathcal{T}}$ avec $\sigma_{\mathcal{T}}$ stratégie totale sur l'arène $\mathbb{F}_0 \to A$ ($\{\varepsilon\}$ étant l'interprétation de

Ω). Dans ce contexte logique, on se restreint de plus à des termes finis en profondeur et donc à des stratégies *finies* qui sont des ensembles finis de vues.

Il est important également, dans l'interprétation de la logique, de contraindre le comportement des stratégies sur les atomes. En effet, si α est un type atomique (*i.e.* $\alpha = \mathbb{F}_k$ dans μPCF), la seule *preuve* de $\alpha \to \alpha$ est l'identité, et par conséquent la seule stratégie « valide » dans $\alpha \to \alpha$ doit être la stratégie copycat.

En considérant les différentes vues possibles sur l'arène $\mathbb{F}_k \to \mathbb{F}_k$, on trouve :



On constate que la première n'est pas rigide arrière et que la deuxième n'est pas rigide avant. Quant à la troisième vue, elle ne correspond à l'identité que si $i = j$, mais ce n'est pas une contrainte structurelle sur les questions et les réponses. Par contre en imposant $k < 2$, le problème est résolu.

On montre ainsi qu'il existe une unique stratégie totale (finie) rigide sur $\mathbb{F}_0 \to \mathbb{F}_0$ et sur $\mathbb{F}_1 \to \mathbb{F}_1$ qui est la stratégie copycat. Le cas de $\mathbb{F}_0$ est particulier puisque dans la mesure où il n'induit aucune réponse, toute stratégie sur $\mathbb{F}_0 \to \mathbb{F}_0$ est rigide ! De manière plus générale, si on considère uniquement des types formés à partir de $\mathbb{F}_0$, on obtient un modèle complet pour le λ-calcul simplement typé avec un seul atome [DHR96], et donc pour la logique minimale avec un seul atome.

Pour résumer, les modèles « logiques » que l'on obtient sont les stratégies totales finies dans les arènes basées sur $\mathbb{F}_0$ comme modèle du λ-calcul, les stratégies totales finies rigides dans les arènes basées sur $\mathbb{F}_1$ comme modèle du $\lambda\mu$-calcul, et les stratégies totales finies rigides bien parenthésées dans les arènes basées sur $\mathbb{F}_1$ comme modèle du λ-calcul. Ces trois modèles sont complets.

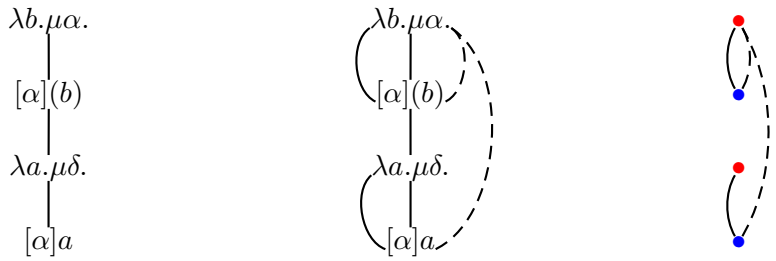
3.2 Logique classique et μ-pointeurs

S'il faut critiquer les trois modèles construits ci-dessus, ceux basés sur $\mathbb{F}_1$ ont le défaut de n'introduire la notion de réponse que pour mieux la contraindre ensuite. Quant à celui sur $\mathbb{F}_0$, il est complet pour le λ-calcul et ne permet donc pas d'interpréter convenablement le $\lambda\mu$-calcul ou la logique classique.

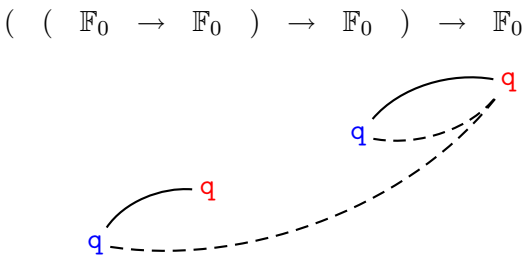
Pour tirer le meilleur parti de ces deux situations, on cherche à ajouter de la structure au modèle basé sur $\mathbb{F}_0$, afin d'avoir « suffisamment de place » pour interpréter le $\lambda\mu$-calcul. On va pour cela utiliser les *μ-pointeurs*. Tout comme pour les λ-pointeurs, chaque coup Joueur d'une vue doit être muni d'un μ-pointeur dont la cible est un coup Opposant précédemment joué. Il n'y a pas d'autre contrainte dans un premier temps (contrairement aux λ-pointeurs qui doivent respecter la structure père-fils de l'arène). Ces μ-pointeurs sont un cas particulier de ceux définis par J. Laird pour modéliser un langage avec exceptions locales [Lai01].

L'interprétation d'un $\lambda\mu$-terme est donnée à partir de sa forme β-normale η-longue (ou *forme canonique*) sous forme d'arbre de Böhm en associant une vue à chaque branche de l'arbre où les liaisons de λ-variables deviennent des λ-pointeurs et les liaisons de μ-variables deviennent des μ-pointeurs. Les formes canoniques du $\lambda\mu$-calcul peuvent être décrites sous la forme $\lambda b_1 \dots \lambda b_k . \mu\beta . [\alpha](a) M_1 \dots M_n$ où les M_i sont en forme canonique. Si on sépare les lieurs

$\lambda b_1 \dots \lambda b_k.\mu\beta.$ vus comme des coups Opposant, des occurrences de variables $[\alpha](a)$ vues comme des coups Joueur, on peut écrire les formes canoniques comme des arbres de coups. Ainsi partant de la forme canonique $\lambda b.\mu\alpha.[\alpha](b)\lambda a.\mu\delta.[\alpha]a : ((\mathbb{F}_0 \to \mathbb{F}_0) \to \mathbb{F}_0) \to \mathbb{F}_0$, on obtient quatre coups. L'arbre de Böhm est obtenu en représentant les liaisons de variables par des λ-arêtes et des μ-arêtes. Enfin, en oubliant la structure interne des coups, on obtient les vues de la stratégie associée au terme de départ.

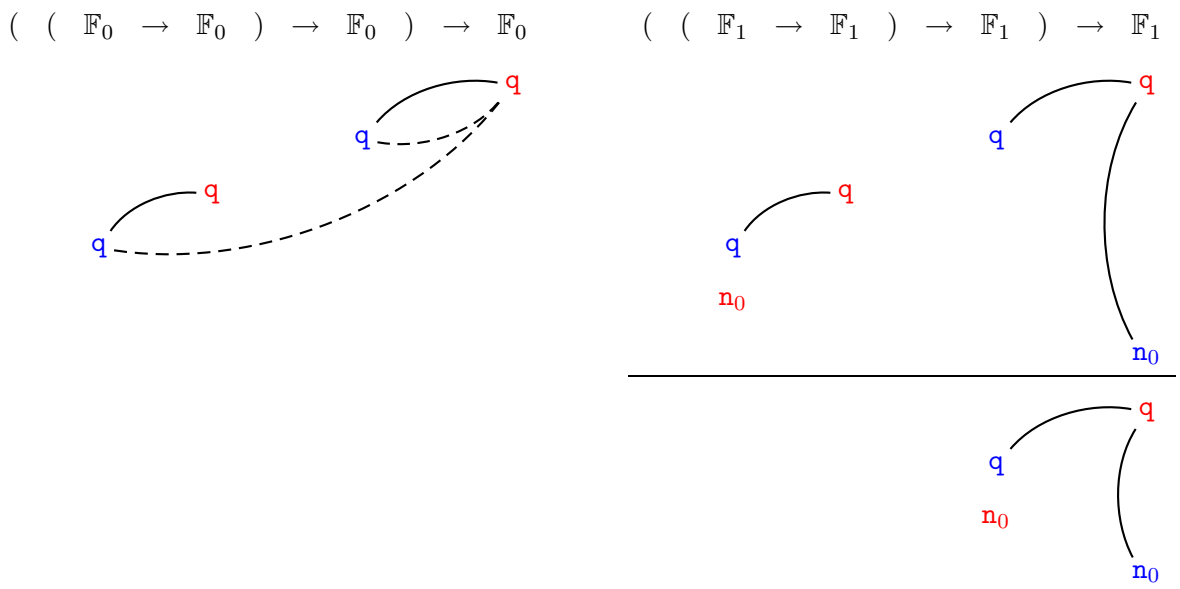


En représentant les coups à la position correspondante dans le type (et donc l'arène), on aboutit à :



Ce modèle avec μ-pointeurs $\mathcal{M}_0$ basé sur $\mathbb{F}_0$ est complet pour le $\lambda\mu$-calcul. Il existe donc nécessairement un rapport étroit avec le modèle $\mathcal{M}_1$ basé sur $\mathbb{F}_1$. Ce rapport s'exprime simplement dans le formalisme des jeux. Étant donné un type, son interprétation dans $\mathcal{M}_0$ est obtenue à partir de celle dans $\mathcal{M}_1$ en supprimant toutes les réponses. Réciproquement, l'interprétation dans $\mathcal{M}_1$ est obtenue à partir de celle dans $\mathcal{M}_0$ en ajoutant une réponse comme fils à chaque nœud. Concernant les termes, on peut définir des procédures de repliage/dépliage permettant de passer de l'interprétation dans $\mathcal{M}_0$ à celle dans $\mathcal{M}_1$. À toute vue de longueur paire dans le modèle $\mathcal{M}_0$, on associe une vue dans $\mathcal{M}_1$ de la manière suivante : si le dernier coup Joueur $\mathtt{m}$ est muni d'un μ-pointeur vers $\mathtt{n}$, on supprime tous les μ-pointeurs et on ajoute deux coups $\mathtt{a_m a_n}$ à la suite obtenue où $\mathtt{a_m}$ est la réponse fille de $\mathtt{m}$ dans l'arène du modèle $\mathcal{M}_1$. Réciproquement, toute vue $\mathtt{s a_m a_n}$ dans $\mathcal{M}_1$ est interprétée comme la présence d'un μ-pointeur de $\mathtt{m}$ vers $\mathtt{n}$ dans $\mathcal{M}_0$.

Dans le cas de l'interprétation de $\lambda b.\mu\alpha.[\alpha](b)\lambda a.\mu\delta.[\alpha]a$, on obtient :



(une vue maximale dans $\mathcal{M}_0$ mais deux dans $\mathcal{M}_1$).

3.3 Logique du premier ordre

Nous n'avons jusqu'ici utilisé qu'un unique type de base (ou une unique formule atomique). Pour le traitement des atomes en logique (et en négligeant pour le moment la quantification du premier ordre), on peut distinguer différents niveaux de richesse :

1. *Propositionnel constant* : on considère uniquement les formules construites à partir des constantes $\top$ (« vrai ») et $\bot$ (« faux ») avec les connecteurs propositionnels. Ainsi les deux seules règles qui permettent de conclure une construction de preuve de bas en haut sont :

$$\frac{}{\Gamma \vdash \top, \Delta}\,\top R \qquad\qquad \frac{}{\bot \vdash}\,\bot L$$

2. *Propositionnel mono-atomique* : on considère uniquement les formules construites à partir des constantes $\top$ et $\bot$, et d'une unique variable propositionnelle R avec les connecteurs propositionnels. La règle « axiome » fait alors sont apparition en haut des preuves :

$$\frac{}{R \vdash R}\,ax$$

   C'est cette règle reliant une occurrence négative d'atome à une occurrence positive d'atome que les μ-pointeurs incarnent.

3. *Propositionnel complet* : on considère un ensemble dénombrable de variables propositionnelles X, Y, ... On obtient des règles « axiome » pour chaque variable propositionnelle et il devient important de les distinguer : $X \vdash Y$ n'étant pas prouvable.

   C'est ici que l'on peut parler de logique propositionnelle. On peut noter en particulier que, pour les formules construites à partir d'une unique variable propositionnelle et du connecteur d'implication, les prouvabilités intuitionniste et classique coïncident. Ce n'est plus le cas en propositionnel complet.

   Ce cas est équivalent au cas Π^1 (formules en forme prénexe avec uniquement des quantifications universelles et du second ordre) car $\forall$ étant un connecteur réversible, on peut le mettre ou l'enlever en début de formule sans réellement modifier la théorie.

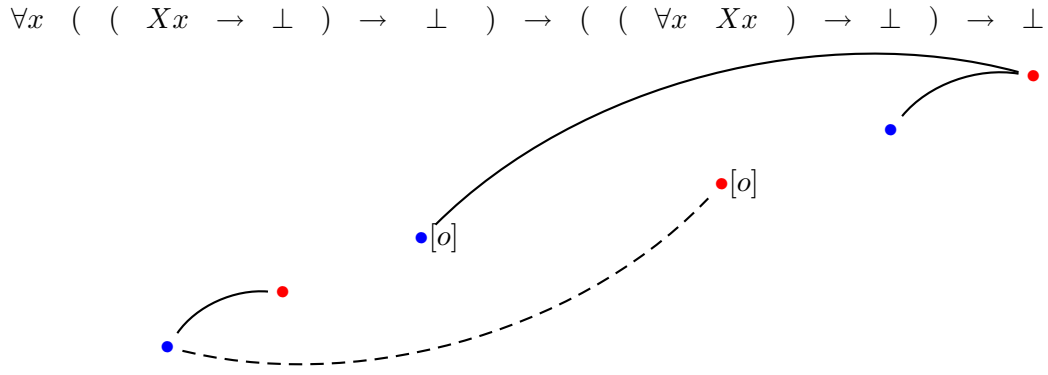
4. *Second ordre* : on introduit les quantificateurs de second ordre en toute généralité, et en particulier la possibilité de substituer les variables propositionnelles par des formules arbitraires lors de la construction de preuve ou l'élimination des coupures. Les variables propositionnelles deviennent véritablement « variables ».

On va décrire comment enrichir notre modèle de jeux pour traiter les systèmes propositionnels complets, puis la quantification du premier ordre [Lau08]. La quantification du second ordre pose des problèmes beaucoup plus complexes dans la mesure où la structure d'arbre des arènes est amenée à évoluer dynamiquement durant les vues [Hug00, dL07] ; nous n'en parlerons pas ici. Sur la base de l'interprétation du type $\mathbb{F}_0$ par l'arène réduite à un unique coup, on interprète l'atome X par un unique coup muni du label X. Ainsi les arènes ont désormais leurs coups décorés par un label (l'absence de label correspondant à $\bot$) ce qui permet de distinguer X de Y par exemple. On demande aux μ-pointeurs de respecter ces labels : les deux extrémités d'un μ-pointeur doivent avoir le même label dans l'arène (une contrainte similaire est introduite dans [Mur01, page 122]). Ainsi seule la première vue ci-dessous est valide (les labels se lisent dans le type au-dessus du coup) :

ce qui correspond à la prouvabilité de $X \vdash X$ et à la non prouvabilité de $X \vdash Y$. En fait on ne demande plus la présence de μ-pointeur que pour les coups Joueur ayant un label (donc ne correspondant pas à $\bot$).

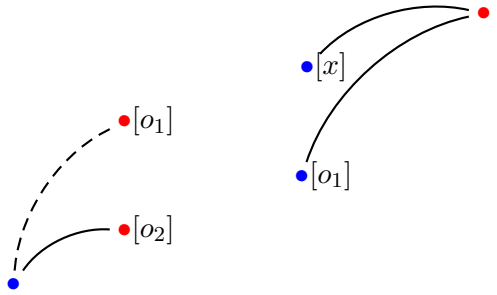
L'interprétation de la quantification du premier ordre se traduit elle aussi par une condition sur les extrémités de μ-pointeurs. Concernant les formules, une formule atomique $Xt_1 \dots t_n$ est interprétée par l'arène à un coup de label $Xt_1 \dots t_n$ et la formule $\forall x A$ est interprétée en ajoutant une nouvelle sorte de label $\forall x$ à la racine de l'interprétation de A. Lorsqu'Opposant joue un coup ayant un label $\forall x$ dans l'arène, il fournit une variable du premier ordre o fraîche et lorsque Joueur joue un tel coup, il fournit un terme t du premier ordre. Ceci permet de définir des substitutions élémentaires $x \mapsto o$ ou $x \mapsto t$. Partant d'une occurrence de coup $\mathtt{m}$ dans une vue, on définit une substitution $\theta_{\mathtt{m}}$ en remontant les λ-pointeurs et en collectant les substitutions élémentaires. Un μ-pointeur d'un coup $\mathtt{m}$ (de label $Xt_1 \dots t_p$ dans l'arène) vers un coup $\mathtt{n}$ (de label $Yt'_1 \dots t'_q$ dans l'arène) est valide si $\theta_{\mathtt{m}}(Xt_1 \dots t_p) = \theta_{\mathtt{n}}(Yt'_1 \dots t'_q)$ (en particulier $X = Y$ et $p = q$). On note ici un interaction forte entre λ-pointeurs, μ-pointeurs et labels dans l'arène.

La commutation de la double négation avec la quantification universelle (qui est valide en logique classique) $\forall x \neg\neg Xx \to \neg\neg \forall x Xx$ peut s'écrire sans négation sous la forme $\forall x((Xx \to \bot) \to \bot) \to ((\forall x Xx) \to \bot) \to \bot$ et admet la stratégie :



La formule dite « du buveur » $\exists x \forall y (Xx \to Xy)$ peut s'écrire sans $\exists$ sous la forme $\forall x(\forall y(Xx \to Xy) \to \bot) \to \bot$ (de manière générale, le connecteur $\exists$ n'étant pas réversible, il ne s'intègre pas directement dans les systèmes en appel par nom, mais passe par le « codage » $\exists \equiv \neg\forall\neg$) et admet la stratégie :

$$\forall x \quad (\quad \forall y \quad (\quad Xx \quad \to \quad Xy \quad) \quad \to \quad \bot \quad) \quad \to \quad \bot$$



Dans le $\lambda\mu$-calcul avec quantification du premier ordre à la Church approprié :

$$M ::= a \mid \lambda a.M \mid (M)M \mid [\alpha]M \mid \mu\alpha.M \mid \Lambda x.M \mid M\{t\}$$

ces stratégies correspondent aux formes canoniques :

$$\lambda b.\lambda a.(a)\Lambda x.\mu\alpha.(b\{x\})\lambda c.[\alpha]c \qquad \text{et} \qquad \lambda b.(b\{x\})\Lambda y.\lambda d.\mu\alpha.(b\{y\})\Lambda z.\lambda a.\mu\delta.[\alpha]a$$

Ce modèle fournit un résultat de complétude pour la logique classique du premier ordre :

Théorème (Équivalence)
La catégorie syntaxique des preuves de logique classique constructive du premier ordre et la catégorie de jeux sont équivalentes.

3.4 Isomorphismes de types

Les modèles de jeux permettent de traiter efficacement la question des isomorphismes de types [Lau05a]. Les isomorphismes syntaxiques sont nécessairement valides dans tout modèle. Le travail sur le modèle consiste alors en deux étapes : caractériser les isomorphismes du modèle puis montrer qu'ils sont tous syntaxiquement valides.

Le point clef est un lemme de « zig-zag » qui montre que deux stratégies σ et τ qui définissent un isomorphisme entre les arènes A et B ont nécessairement la propriété d'alterner les coups dans A et les coups dans B en transportant la structure (λ-pointeurs, μ-pointeurs, termes du premier ordre) d'un côté à l'autre. Ceci mène à une caractérisation « géométrique » :

Théorème (Isomorphismes d'arènes)
Deux arènes sont isomorphes dans le modèle de jeux si et seulement si elle le sont en tant qu'arbres étiquetés.

Par une récurrence sur la taille de ces arbres, on extrait les équations entre types/formules

qui correspondent :

$$
\begin{aligned}
A \vee B &= B \vee A \\
A \vee (B \vee C) &= (A \vee B) \vee C \\
A \vee \bot &= A
\end{aligned}
$$

$$
\begin{aligned}
A \wedge B &= B \wedge A \\
A \wedge (B \wedge C) &= (A \wedge B) \wedge C \\
A \wedge \top &= A
\end{aligned}
$$

$$
\forall x \forall y A = \forall y \forall x A
$$

$$
\begin{aligned}
A \vee (B \wedge C) &= (A \vee B) \wedge (A \vee C) \\
A \vee \top &= \top \\
A \vee \forall x B &= \forall x (A \vee B) \qquad\qquad x \notin A
\end{aligned}
$$

$$
\begin{aligned}
\forall x (A \wedge B) &= \forall x A \wedge \forall x B \\
\forall x \top &= \top
\end{aligned}
$$

$$
\begin{aligned}
\neg (A \wedge B) &= \neg A \vee \neg B \\
\neg \top &= \bot
\end{aligned}
$$

$$
A \to B = \neg A \vee B
$$

La sémantique des jeux permet ainsi de résoudre élégamment un problème d'origine purement syntaxique.

Les résultats présentés ici semblent clore la question du rapport entre logique classique constructive du premier ordre et sémantique des jeux. Après le passage de la logique intuitionniste propositionnelle à la logique classique constructive propositionnelle puis à logique classique constructive du premier ordre, on souhaite être en mesure de traiter le prédicat d'égalité, d'intégrer les travaux sur la logique du second ordre [dL07], d'interpréter des axiomes comme l'axiome du choix, ...

Une autre direction plus prospective consiste à trouver d'autres domaines sémantiques que les modèles de jeux pour lesquels on puisse prouver des théorèmes d'équivalence et qui fournissent donc d'autres présentations des formes canoniques de la logique classique constructive.

Deuxième partie

Logiques linéaires allégées et complexité

[LTdF06] Olivier Laurent and Lorenzo Tortora de Falco. Obsessional cliques: a semantic characterization of bounded time complexity. In *Proceedings of the twenty-first annual symposium on Logic In Computer Science*, pages 179–188, Seattle, August 2006. IEEE, IEEE Computer Society Press.

[DLL08] Ugo Dal Lago and Olivier Laurent. Quantitative game semantics for linear logic. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic*, volume 5213 of *Lecture Notes in Computer Science*, pages 230–245. Springer, September 2008.

[Lau09] Olivier Laurent. On the categorical semantics of elementary linear logic. *Theory and Applications of Categories*, 22(10):269–301, June 2009.

La logique linéaire a permis de mettre en avant la notion d'utilisation de ressource à l'intérieur même de la logique : une formule n'est par défaut utilisable qu'un nombre spécifié de fois et les connecteurs exponentiels stipulent explicitement la possibilité de passer outre. De cet aspect « qualitatif » (une formule est réutilisable ou ne l'est pas), il est possible de passer à des éléments « quantitatifs » en limitant le nombre d'étapes d'élimination des coupures par des restrictions appropriées des connecteurs exponentiels (on parle de *logiques linéaires allégées*). Ainsi les systèmes LLL [Gir98] ou SLL [Laf04] représentent la complexité en temps polynomial et ELL [Gir98, DJ03] la complexité en temps élémentaire. Il s'agit d'une approche de la complexité dite *implicite*, car aucun artefact externe ne sert à borner le temps de calcul, c'est la structure des objets eux-mêmes qui garantit les bornes. Étant donnés un système et une notion de représentation des prédicats sur les entiers (essentiellement des preuves représentant les booléens et les entiers, et une fonction associant, à tout prédicat, une preuve qui, composée avec la représentation d'un entier, se normalise en le booléen que vaut le prédicat en cet entier), on dit qu'on a *correction* vis-à-vis d'une classe de complexité si tous les prédicats représentables appartiennent à cette classe (ce qui implique des contraintes sur la complexité de la normalisation dans le système logique), et on a *complétude extensionnelle* si tous les prédicats de la classe sont représentables. Ainsi LLL et SLL sont corrects et extensionnellement complets pour le temps polynomial et ELL pour le temps élémentaire, à condition de considérer ces systèmes avec la quantification du second ordre (les connecteurs additifs ne sont par contre pas nécessaires [MT03]). Il faut noter que la complétude extensionnelle ne dit pas tout : si toute fonction calculable en temps polynomial admet un algorithme qui permet de la représenter, dans SLL par exemple, tout algorithme en temps polynomial n'admet pas nécessairement de représentation directe dans SLL. On oppose ainsi l'expressivité extensionnelle en termes de fonctions représentables (dont parle la complétude extensionnelle) et l'*expressivité intentionnelle* en termes d'algorithmes représentables (qui diffère entre LLL et SLL par exemple). Concernant la correction, les résultats de complexité de l'élimination des coupures obtenus dans le cadre des logiques linéaires allégées sont souvent forts : alors que les propriétés de terminaison des calculs sont généralement liés à l'utilisation explicite des types ou formules (en λ-calcul, en logique linéaire complète, ...), on peut prouver les résultats de complexité de ces fragments allégés de LL indépendamment des types, sur la seule base d'invariants graphiques sur les boîtes des réseaux de preuve.

Les trois systèmes qui vont nous intéresser ici sont ELL, LLL et SLL. Dans l'objectif de mieux comprendre les structures sous-jacentes à ces logiques, on adopte une approche sémantique. C'est une méthodologie courante dans le domaine : la logique linéaire est issue des espaces cohérents, LC des espaces de corrélation, la logique linéaire différentielle des sémantiques vectorielles, ...

Afin de donner un cadre à cette étude sémantique, on s'intéresse tout d'abord à la définition d'une notion catégorique de modèle (principalement pour ELL). De plus la preuve que la notion introduite (catégorie de Seely élémentaire par exemple) fournit bien des modèles (de ELL) facilite ensuite la preuve de correction de modèles particuliers : prouver que $\mathcal{M}$ est un modèle de ELL se factorise en prouvant que $\mathcal{M}$ est une catégorie de Seely élémentaire (ce qui réduit le travail, notamment en présence de connecteurs additifs), dans la mesure où on a prouvé une fois pour toutes qu'une telle catégorie est toujours un modèle de ELL.

Sur la base du modèle relationnel (le modèle le plus simple de la logique linéaire [BE01]), en définissant une propriété de saturation des interprétations de preuves, on introduit deux modèles concrets : l'un pour ELL et l'autre pour SLL. Une forme relativisée de complétude montre que ces modèles permettent de caractériser ELL et SLL parmi les preuves de LL.

Cette première approche consiste essentiellement à restreindre un modèle existant de LL pour coller au mieux à un système allégé donné (et donc à une classe de complexité). Une alternative est de considérer un modèle de LL dans lequel on puisse exprimer des bornes de

complexité : un modèle muni d'une fonction $\mathcal{C}$ à valeurs entières telle que $\mathcal{C}(\llbracket \pi \rrbracket)$ donne une borne sur le temps de normalisation de π. On utilise pour cela un modèle de jeux, profitant une fois de plus de la pertinence de ces modèles pour analyser la syntaxe.

4 Trois logiques allégées

Parmi les différents sous-systèmes de LL à complexité bornée, nous nous intéresserons principalement à ELL et SLL (et ponctuellement à LLL). Comme nous le mentionnions précédemment, les résultats de complétude extensionnelle nécessitent la quantification du second ordre. Nous ne parlerons cependant ici que des fragments propositionnels des différents systèmes.

4.1 Règles exponentielles

Commençons par les fragments multiplicatifs et exponentiels. Les règles multiplicatives restent celles de LL (on pourra se reporter à l'annexe pour trouver un rappel des règles des différents systèmes étudiés). Elles induisent un comportement linéaire dans l'élimination des coupures et n'ont donc pas d'impact sur la complexité. La présentation originelle de la règle de promotion par J.-Y. Girard n'est pas la plus adaptée pour les systèmes allégés. On s'intéresse donc aux règles exponentielles suivantes (dites à base de *promotion multi-fonctorielle*) pour LL :

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

La règle de « digging » (??) est souvent critiquée sur le plan syntaxique pour ce qu'elle viole la propriété de la sous-formule. Cependant elle apparaît naturellement dans le contexte des modèles catégoriques (en lien avec la notion de monade). En ce qui nous concerne ici, on peut ignorer cette discussion sur les propriétés intrinsèques du digging puisque cette règle va être retirée pour définir ELL, LLL et SLL.

Logique linéaire élémentaire. Le système ELL est obtenu en supprimant les règles de déréliction ($?d$) et de digging (??), il reste :

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

Logique linéaire douce. Le système SLL ne conserve que la règle de promotion multi-fonctorielle et substitue, aux autres règles, la règle de « multiplexing », on a donc :

$$\frac{\vdash \Gamma, A, \ldots, A}{\vdash \Gamma, ?A}\,?m \qquad \frac{\vdash \Gamma, A}{\vdash ?\Gamma, !A}\,!f$$

La règle de multiplexing ($?m$) admet comme cas particuliers la règle de déréliction et la règle d'affaiblissement de la logique linéaire, mais pas la règle de contraction. En effet, ajouter la règle de contraction à SLL en fait un système plus puissant que ELL et brise donc la borne de complexité polynomiale sur la normalisation de SLL. Réciproquement, la règle de multiplexing est dérivable dans LL.

Logique linéaire légère. Le système LLL contraint, dans ELL, la règle de promotion à n'avoir qu'une formule en contexte (on parle de *promotion fonctorielle* et non multi-fonctorielle). Le

pouvoir expressif du système obtenu étant trop restreint, il est nécessaire d'introduire une nouvelle paire de connecteurs duaux § et $\bar{\S}$. Les règles exponentielles de LLL sont ainsi :

$$\frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\,?c \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\,?w \qquad \frac{\vdash B, A}{\vdash ?B, !A}\,!f \qquad \frac{\vdash \Gamma, \Delta, A}{\vdash ?\Gamma, \bar{\S}\Delta, \S A}\,\S f$$

Les deux connecteurs exponentiels ! et § sont alors reliés par le principe $!A \vdash \S A$.

4.2 Connecteurs additifs

Rajouter aux systèmes ci-dessus les règles des connecteurs additifs de la logique linéaire ne pose pas de problème particulier et n'a notamment pas d'impact sur les questions de complexité.

Les choses se compliquent si l'on s'intéresse à l'interaction entre connecteurs additifs et exponentiels (probablement l'un des points les moins clairs à ce jour dans la théorie de la logique linéaire). Une des propriétés centrales de LL reliant les différents types de connecteurs est l'isomorphisme $!(A \,\&\, B) \simeq !A \otimes !B$. Les modifications apportées aux règles exponentielles dans ELL, LLL et SLL brisent cet isomorphisme clef.

Nous allons voir qu'il n'est pas raisonnable de ré-introduire cet isomorphisme dans SLL, mais que cela s'avère possible dans ELL (la méthode s'appliquant de manière similaire à LLL). En présence de promotion fonctorielle et de connecteurs additifs (donc d'une diagonale $A \xrightarrow{\Delta} A \,\&\, A$), l'isomorphisme ci-dessus rend la contraction dérivable : $!A \xrightarrow{!\Delta} !(A \,\&\, A) \simeq !A \otimes !A$ (or on a déjà vu que ce n'était pas acceptable dans SLL). Dans le cas de ELL, V. Danos et J.-B. Joinet [DJ03] ont montré comment retrouver l'isomorphisme clef sans perdre la normalisation en temps élémentaire. Leur solution peut se présenter en calcul des séquents à l'aide un *pré-connecteur* unaire $\flat$. Ainsi $\flat A$ est une *pré-formule* si A est une formule ne contenant pas $\flat$, et les (autres) connecteurs ne s'appliquent qu'à des formules (pas des pré-formules) : $\flat\flat A$ et $\flat A \otimes 1$ ne sont ni des formules ni des pré-formules. Le pré-connecteur $\flat$ agit comme un marqueur sur les formules. Une dérivation dont la conclusion contient des pré-formules est alors une *pré-preuve*, le terme *preuve* étant réservé aux dérivations dont la conclusion ne contient que des formules (donc sans $\flat$).

Les règles exponentielles de ELL deviennent :

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

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

En l'absence de règles additives, ces règles sont équivalentes à celles décrites précédemment : les règles $(\flat)$, $(\flat c)$ et $(\flat w)$ commutent vers le bas avec toutes les autres règles excepté $(!\flat)$; $(\flat c)$ et $(\flat w)$ commutent vers le bas avec $(!\flat)$ en devenant $(?c)$ et $(?w)$; enfin dans une preuve (pas nécessairement dans une pré-preuve) les règles $(\flat)$ se trouvent alors agglomérées au-dessus des règles $(!\flat)$, se transformant ensemble en des règles $(!f)$. En présence de règles additives et de $\flat$, l'isomorphisme $!(A \,\&\, B) \simeq !A \otimes !B$ redevient prouvable :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^\perp, A}\,ax}{\vdash \flat A^\perp, A}\,\flat}{\vdash \flat A^\perp, \flat B^\perp, A}\,\flat w \qquad \dfrac{\dfrac{\dfrac{}{\vdash B^\perp, B}\,ax}{\vdash \flat B^\perp, B}\,\flat}{\vdash \flat A^\perp, \flat B^\perp, B}\,\flat w}{\vdash \flat A^\perp, \flat B^\perp, A \,\&\, B}\,\&}{\vdash ?A^\perp, ?B^\perp, !(A \,\&\, B)}\,!\flat}{\vdash ?A^\perp \parr ?B^\perp, !(A \,\&\, B)}\,\parr$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A^\perp, A}\,ax}{\vdash A^\perp \oplus B^\perp, A}\,\oplus_1}{\vdash \flat(A^\perp \oplus B^\perp), A}\,\flat}{\vdash ?(A^\perp \oplus B^\perp), !A}\,!\flat \qquad \dfrac{\dfrac{\dfrac{}{\vdash B^\perp, B}\,ax}{\vdash A^\perp \oplus B^\perp, B}\,\oplus_2}{\vdash \flat(A^\perp \oplus B^\perp), B}\,\flat}{\vdash ?(A^\perp \oplus B^\perp), !B}\,!\flat}{\vdash ?(A^\perp \oplus B^\perp), ?(A^\perp \oplus B^\perp), !A \otimes !B}\,\otimes}{\vdash ?(A^\perp \oplus B^\perp), !A \otimes !B}\,?c$$

On peut noter que la traduction $\flat A \mapsto ?A$ permet de plonger ce système avec $\flat$ dans LL, et que les deux preuves ci-dessus redonnent ainsi celles de l'isomorphisme $!(A \,\&\, B) \simeq !A \otimes !B$ dans LL.

Pouvoir étendre les logiques linéaires allégées sans briser les bornes de complexité associées est un point important dans la recherche de systèmes permettant de représenter le plus possible d'algorithmes (*i.e.* allant vers plus d'expressivité intentionnelle). L'ajout des connecteurs additifs les plus expressifs possible va dans ce sens. On s'intéresse souvent aussi à l'ajout de l'affaiblissement dans les variantes intuitionnistes des logiques allégées [AR02].

5 Modèles catégoriques

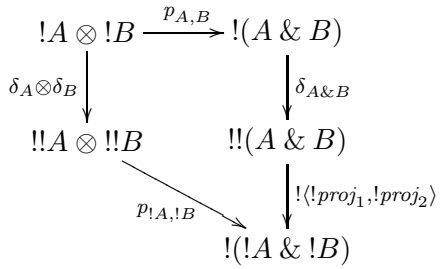
L'étude des modèles dénotationnels d'un système logique passe souvent par une définition catégorique de ces modèles.

5.1 Modèles à la Seely

Dans le cas de LL, la première proposition d'une telle définition est due à R. Seely [See89] (on s'intéresse en fait directement à la version corrigée par G. Biermann [Bie95, Mel03] et de plus au cas $\star$-autonome et pas seulement monoïdal clos).

Définition (Catégorie de Seely)
Une *catégorie de Seely* $\mathcal{L}$ est une catégorie $\star$-autonome avec produits finis munie d'un endofoncteur ! tel que :

1. $(!, \delta, \varepsilon)$ est une comonade
2. $(!, p, q)$ est un foncteur monoïdal symétrique fort de $(\mathcal{L}, \&, \top)$ dans $(\mathcal{L}, \otimes, 1)$
3. le diagramme suivant commute :



La suppression des règles de déréliction et digging lors du passage de LL à ELL correspond naturellement à la suppression de la structure de comonade de ! (et par conséquent le diagramme de la condition 3 n'a plus de raison d'être). Il est cependant nécessaire de rajouter de nouvelles conditions, la disparition de la structure de comonade faisant perdre trop. En particulier, dans une catégorie de Seely, il découle de la structure présente que $(!, m, n)$ est un foncteur monoïdal symétrique de $(\mathcal{L}, \otimes, 1)$ dans $(co\mathcal{MON}(\mathcal{L}), \otimes, 1)$ (où $co\mathcal{MON}(\mathcal{L})$ est la catégorie des $\otimes$-comonoïdes de $\mathcal{L}$). On arrive ainsi à la définition suivante de modèle de ELL [Lau09] :

Définition (Catégorie de Seely élémentaire)
Une *catégorie de Seely élémentaire* $\mathcal{C}$ est une catégorie $\star$-autonome avec produits finis munie d'un endofoncteur ! tel que :

1. $(!, m, n)$ est un foncteur monoïdal symétrique de $(\mathcal{C}, \otimes, 1)$ dans $(co\mathcal{MON}(\mathcal{C}), \otimes, 1)$
2. $(!, p, q)$ est un foncteur monoïdal symétrique fort de $(\mathcal{C}, \&, \top)$ dans $(\mathcal{C}, \otimes, 1)$
3. ! transporte la structure de &-comonoïde de A sur la structure de $\otimes$-comonoïde de $!A$

Comme suggéré ci-dessus, on montre que toute catégorie de Seely est une catégorie de Seely élémentaire.

Une fois cette définition donnée, se pose la question de l'interprétation d'un système logique avec pré-connecteur, comme l'est l'extension de ELL avec connecteurs additifs. Le schéma classique en logique catégorique consiste à établir les correspondances suivantes : formule $\mapsto$ objet, preuve $\mapsto$ morphisme, connecteur $\mapsto$ foncteur (ou multi-foncteur), et règle $\mapsto$ transformation naturelle. Pour simplifier les notations, notons $\sharp$ le dual de $\flat$ ($\sharp A = (\flat A^\perp)^\perp$) qui est à ! ce que $\flat$ est à ?. On ne va pas interpréter $\sharp$ par un foncteur mais, de manière plus générale, par toute une famille de foncteurs $\mathcal{S}$. Ainsi une pré-preuve π de conclusion $\vdash \flat A^\perp, B$ sera interprétée par un morphisme de $\sharp A$ dans B où $\sharp$ est un élément de $\mathcal{S}$ *qui dépend de* π. Dans le cas d'une preuve (et pas d'une pré-preuve), les pré-connecteurs disparaissent et les objets source et cible du morphisme interprétant la preuve ne dépendent plus de celle-ci. On retrouve le cadre usuel.

On définit $\mathcal{S}$ comme la plus petite famille d'endofoncteurs de $\mathcal{C}$ contenant le foncteur identité et les foncteurs constants 1 et $\top$, et close par $\otimes$ et par & (ainsi $A \mapsto \top \otimes ((A \otimes A) \,\&\, (1 \,\&\, A))$ est un élément de $\mathcal{S}$). On montre que tout élément de $\mathcal{S}$ est un foncteur monoïdal symétrique de $(\mathcal{C}, \otimes, 1)$ dans $(\mathcal{C}, \otimes, 1)$, ce qui fait de $!\sharp$ un foncteur monoïdal symétrique de $(\mathcal{C}, \otimes, 1)$ dans $(co\mathcal{MON}(\mathcal{C}), \otimes, 1)$ (dès que $\sharp \in \mathcal{S}$). On définit de plus une transformation naturelle monoïdale $b^\sharp$ de ! dans $!\sharp$ pour chaque $\sharp$ dans $\mathcal{S}$. Ceci fournit les ingrédients nécessaires à l'interprétation de ELL. À titre d'exemple, voici comment interpréter les règles ($\flat c$) et ($!\flat$) dans une catégorie de Seely élémentaire :

($\flat c$) Un morphisme de $\sharp_1(A) \otimes \sharp_2(A)$ dans B est un morphisme de $\sharp_1 \otimes \sharp_2(A)$ dans B avec $\sharp_1 \otimes \sharp_2 \in \mathcal{S}$ si $\sharp_1 \in \mathcal{S}$ et $\sharp_2 \in \mathcal{S}$.

($!\flat$) Si $\sharp_1$ et $\sharp_2$ sont deux éléments de $\mathcal{S}$ et si f est un morphisme de $\sharp_1 A \otimes \sharp_2 B$ dans C, on construit :

$$!A \otimes !B \xrightarrow{b^{\sharp_1} \otimes b^{\sharp_2}} !\sharp_1 A \otimes !\sharp_2 B \xrightarrow{m} !(\sharp_1 A \otimes \sharp_2 B) \xrightarrow{!f} !C$$

Juste un mot concernant LLL : la définition d'une notion de *catégorie de Seely légère*, en supprimant la condition de monoïdalité de ! de $(\mathcal{C}, \otimes, 1)$ dans $(co\mathcal{MON}(\mathcal{C}), \otimes, 1)$, en introduisant un foncteur monoïdal symétrique § de $(\mathcal{C}, \otimes, 1)$ dans $(\mathcal{C}, \otimes, 1)$ et en demandant l'existence d'une transformation naturelle de ! dans § (pour interpréter $!A \vdash \S A$), ne doit pas poser de problème et ne doit apporter aucune nouveauté particulière par rapport au cas de ELL.

Pour ce qui est de SLL, le travail a été fait par B. Redmond [Red07] qui a défini les catégories à multiplexeur.

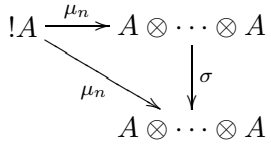
Définition (Catégorie à multiplexeur)
Une *catégorie à multiplexeur* $\mathcal{M}$ est une catégorie $\star$-autonome munie d'un endofoncteur ! tel que :

1. $(!, m, n)$ est un foncteur monoïdal symétrique de $(\mathcal{M}, \otimes, 1)$ dans $(\mathcal{M}, \otimes, 1)$
2. pour tout entier n, il existe une transformation naturelle monoïdale μ_n de ! dans $A \mapsto A \otimes \cdots \otimes A$ (n fois)
3. les deux diagrammes suivants commutent:

$$\begin{array}{ccc} !A \otimes !B & \xrightarrow{\quad m \quad} & !(A \otimes B) \\ {\scriptstyle \mu_n \otimes \mu_n}\downarrow & & \downarrow{\scriptstyle \mu_n} \\ A \otimes \cdots \otimes A \otimes B \otimes \cdots \otimes B & \xrightarrow[\sim]{} & (A \otimes B) \otimes \cdots \otimes (A \otimes B) \end{array} \qquad \begin{array}{ccc} 1 & \xrightarrow{\quad n \quad} & !1 \\ & \searrow^{\sim} & \downarrow{\scriptstyle \mu_n} \\ & & 1 \otimes \cdots \otimes 1 \end{array}$$

Ajouter l'hypothèse que $\mathcal{M}$ a des produits finis suffit à interpréter les règles usuelles des connecteurs additifs de LL.

Une catégorie à multiplexeur est dite *symétrique* si le diagramme suivant commute :



pour toute permutation σ.

5.2 Modèles linéaires/non-linéaires

Une proposition alternative concernant la définition de modèles catégoriques de LL est due à N. Benton [Ben94] (à nouveau on donne la version $\star$-autonome et avec produits).

Définition (Modèle linéaire/non-linéaire)
Un *modèle linéaire/non-linéaire* est donné par une catégorie $\star$-autonome $\mathcal{C}$ avec produits finis, une catégorie cartésienne $\mathcal{M}$ et une adjonction monoïdale symétrique entre elles.

Un modèle linéaire/non-linéaire fournit donc un foncteur F de $\mathcal{C}$ dans $\mathcal{M}$ adjoint à droite d'un foncteur G de $\mathcal{M}$ dans $\mathcal{C}$, tous deux monoïdaux symétriques.

Le foncteur ! est obtenu comme la composition $G \circ F$. Sa structure de comonade (donc la possibilité d'interpréter les règles de déréliction et de digging) est induite par l'adjonction. Comme avec les catégories de Seely, supprimer brutalement la propriété d'adjonction pour définir une version élémentaire est trop violent. Il est nécessaire d'exiger certaines propriétés qui découlaient de cette d'adjonction. On arrive à la définition suivante :

Définition (Modèle linéaire/non-linéaire élémentaire)
Un *modèle linéaire/non-linéaire élémentaire* est donné par une catégorie $\star$-autonome $\mathcal{C}$ avec produits finis, une catégorie cartésienne $\mathcal{M}$, un foncteur F de $\mathcal{C}$ dans $\mathcal{M}$ et un foncteur G de $\mathcal{M}$ dans $\mathcal{C}$ tels que :

1. F est un foncteur monoïdal symétrique
2. G est un foncteur monoïdal symétrique fort
3. F préserve les produits

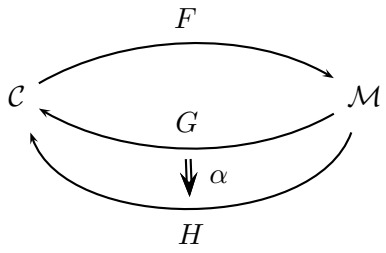
Tout modèle linéaire/non-linéaire est un modèle linéaire/non-linéaire élémentaire.

On montre que les notions de catégorie de Seely élémentaire et de modèle linéaire/non-linéaire élémentaire sont équivalentes au sens où : toute catégorie de Seely élémentaire induit un modèle linéaire/non-linéaire élémentaire et tout modèle linéaire/non-linéaire élémentaire induit une catégorie de Seely élémentaire. Ceci implique que les modèles linéaires/non-linéaires élémentaires permettent d'interpréter ELL.

La définition de modèles de LLL nécessite d'introduire un foncteur § tel que $\S A$ ne soit pas muni d'une structure de $\otimes$-comonoïde. Dans le contexte linéaire/non-linéaire, ceci interdit de composer F avec un foncteur op-monoïdal. De même, rendre ! non monoïdal interdit de composer F avec un foncteur monoïdal. On introduit donc deux foncteurs différents de $\mathcal{M}$ dans $\mathcal{C}$, pour aboutir à la proposition suivante de modèle de LLL :

Définition (Modèle linéaire/non-linéaire léger)
Un *modèle linéaire/non-linéaire léger* est donné par une catégorie $\star$-autonome $\mathcal{C}$ avec produits finis, une catégorie cartésienne $\mathcal{M}$, un foncteur F de $\mathcal{C}$ dans $\mathcal{M}$, deux foncteurs G et H de $\mathcal{M}$ dans $\mathcal{C}$ et une transformation naturelle α de G dans H tels que :

1. F est un foncteur monoïdal symétrique
2. G est un foncteur op-monoïdal symétrique
3. H est un foncteur monoïdal symétrique
4. F préserve les produits



La transformation naturelle α permet d'interpréter $!A \vdash \S A$ en définissant $! = G \circ F$ et $\S = H \circ F$.

Tout modèle linéaire/non-linéaire élémentaire est un modèle linéaire/non-linéaire léger : c'est le cas particulier où $G = H$ est monoïdal fort et où $\alpha = id$.

La pertinence de ces modèles de LLL reste cependant à étudier.

L'approche linéaire/non-linéaire ne semble pas adaptée à SLL, où les $\otimes$-comonoïdes ne sont pas prépondérants contrairement à LL, ELL et LLL.

6 Le modèle obsessionnel

Une fois le cadre catégorique établi, on cherche à construire des modèles concrets. Les modèles dénotationnels des logiques linéaires allégées sont encore toutefois peu nombreux dans la littérature [Bai04, MO00, DLH05].

Une approche naturelle consiste à chercher des propriétés sémantiques au sein des modèles de LL qui correspondent aux spécificités des systèmes allégés. L'intérêt est alors de construire des modèles qui réfutent les principes responsables de l'explosion de complexité qui a lieu dans LL. Ainsi un modèle de ELL ne doit pas valider la déréliction $!A \multimap A$ et un modèle de SLL ne doit pas valider la contraction $!A \multimap !A \otimes !A$. Mieux encore, une approche aboutie de la sémantique de ces systèmes devrait permettre de déduire les bornes de complexité directement des modèles sans retour à la syntaxe. Des travaux dans cette direction sont en cours.

C'est cette volonté de trouver des modèles des systèmes allégés qui ne soient pas des modèles de LL tout entier et qui puissent donc être significatifs sur le plan de la complexité qui ont guidé les travaux mentionnés précédemment sur les modèles catégoriques. L'adéquation du travail catégorique abstrait avec la logique est conforté par le fait que les deux principaux modèles de ELL (dont nous allons parler maintenant), le modèle stratifié et le modèle obsessionnel, fournissent tous les deux des catégories de Seely élémentaires.

La première proposition dans le domaine de la sémantique des systèmes allégés est celle des espaces cohérents stratifiés [Bai04] qui définissent des modèles de ELL et LLL. Elle formalise au niveau sémantique la propriété de *stratification* satisfaite par ces deux systèmes : la profondeur d'un nœud dans un réseau ne change pas durant l'élimination des coupures. L'interprétation de la règle de promotion/boîte dans les sémantiques relationnelle et cohérente se fait en collectant un nombre arbitraire d'interprétations de la prémisse sous forme de multi-ensemble. Ceci est particulièrement clair lorsqu'on utilise le calcul de la sémantique par les expériences [Gir87]. En associant les expériences obsessionnelles [TdF03] (une forme particulière d'expérience répétant toujours, et un même nombre de fois, les calculs effectués dans la prémisse d'une promotion/boîte, et qui fournissent ainsi un outil particulièrement utile pour coder de l'information sur la structure d'un réseau) et l'idée de stratification, on aboutit à une autre notion, les *cliques obsessionnelles* : si la profondeur d'un nœud est un invariant de la réduction alors, partant d'un élément de $[\![\pi]\!]$, il doit être possible d'en obtenir un autre en itérant un plus grand nombre de fois ce qui se passe dans les boîtes. Cliques stratifiées et cliques obsessionnelles fournissent ainsi deux formulations sémantiques assez différentes de la propriété de stratification

de ELL. En particulier, on sait adapter les cliques stratifiées pour modéliser LLL mais pas pour SLL, alors qu'on sait adapter les cliques obsessionnelles pour modéliser SLL mais pas aussi bien pour LLL.

Le modèle des cliques obsessionnelles [LTdF06] est défini en munissant les ensembles utilisés dans le modèle relationnel (on pourrait également travailler sur le modèle cohérent [Lau09]) d'une action du monoïde multiplicatif des entiers non nuls $(\mathbb{N}^\star, \times, 1)$. Dans le modèle relationnel, une formule A est interprétée par un ensemble $[\![A]\!]$ (la trame) et les preuves π de A par des sous-ensembles $[\![\pi]\!]$ de $[\![A]\!]$. Les connecteurs multiplicatifs $\otimes$, $\parr$ et $\multimap$ sont interprétés par le produit cartésien, les connecteurs additifs & et $\oplus$ par l'union disjointe et enfin $[\![!A]\!] = [\![?A]\!] = \mathcal{M}_{fin}([\![A]\!])$ (où $\mathcal{M}_{fin}(E)$ est l'ensemble des multi-ensembles finis d'éléments de E). Concernant l'action du monoïde $(\mathbb{N}^\star, \times, 1)$, elle est donnée composante par composante pour les connecteurs multiplicatifs (et additifs). Le cas intéressant est celui des connecteurs exponentiels. Si t est un entier, on définit une famille (indicée par $t \in \mathbb{N}$) d'actions sur $[\![!A]\!]$ à partir de l'action de $[\![A]\!]$:

$$k \cdot_t [a_1, \ldots, a_n] = \begin{cases} [k \cdot_t a_1, \ldots, k \cdot_t a_n] & \text{si } n \leq t \\ [k \cdot_t a_1, \ldots, k \cdot_t a_n]^k & \text{si } n > t \end{cases}$$

où $[a_1, \ldots, a_n]^k$ est le multi-ensemble à kn éléments obtenu en copiant k fois chaque élément. On dit alors que $c \subseteq [\![A]\!]$ est t-obsessionnelle si, pour tout $x \in c$ et pour tout $k \in \mathbb{N}^\star$, $k \cdot_t x \in c$. On utilise ici la terminologie de *clique* car les constructions proposées s'appliquent à l'identique dans le cas du modèle cohérent.

On montre que si π est une preuve de ELL alors $[\![\pi]\!]$ est 0-obsessionnelle et si π est une preuve de SLL alors $[\![\pi]\!]$ est t-obsessionnelle pour tout t suffisamment grand. Ainsi la déréliction interprétée comme $\mathtt{d} = \{([a], a) \mid a \in [\![A]\!]\}$ est t-obsessionnelle exactement pour $t > 0$ (dès qu'il y a copie à l'intérieur du multi-ensemble, on sort de $\mathtt{d}$) en accord avec le fait qu'elle est valide dans SLL mais pas dans ELL. D'autre part, la contraction interprétée comme $\mathtt{c} = \{(\mu + \nu, \mu, \nu) \mid (\mu, \nu) \in [\![!A]\!]^2\}$ est 0-obsessionnelle et acceptée par ELL mais n'est obsessionnelle pour aucun autre indice et donc bien rejetée pour SLL (sinon il peut y avoir copie dans la somme mais pas dans les composants pris séparément et on sort de $\mathtt{c}$). L'introduction du paramètre t variable pour l'interprétation de SLL est très similaire à la construction catégorique générale introduite indépendamment par B. Redmond [Red07].

Du travail général sur les modèles catégoriques de ELL découle une preuve très simple du fait que le modèle des cliques obsessionnelles s'étend aux connecteurs additifs pour ELL. Dans la mesure où l'on travaille à l'intérieur d'un modèle de LL, il suffit de montrer que les constructions nécessaires à l'interprétation des preuves de ELL préservent l'obsessionalité (les égalités requises entre morphismes découlent directement du fait que le modèle de LL ambiant les valide). En l'absence de connecteurs additifs, il suffit de vérifier chaque règle du calcul des séquents. Pour les connecteurs additifs, l'aspect global de leur intégration nécessite d'isoler les principes nécessaires à l'interprétation des preuves, ce que l'analyse catégorique a permis : il suffit de vérifier que & fournit un produit cartésien dans la catégorie obsessionnelle et que l'isomorphisme $!(A \,\&\, B) \simeq !A \otimes !B$ est obsessionnel.

6.1 Complétude relative

Nous nous sommes intéressés à la complétude forte des modèles ainsi construits. La réponse est immédiatement négative : ELL et SLL contiennent tous les deux MLL pour lequel le modèle relationnel n'est pas complet et l'obsessionalité inopérante. Ceci conduit à l'introduction d'une notion plus faible de *complétude relative*. Si $\mathcal{S}_0$ est un sous-système de $\mathcal{S}$ et si $\mathcal{M}_0$ est un modèle de $\mathcal{S}_0$ obtenu en restreignant un modèle $\mathcal{M}$ de $\mathcal{S}$, $\mathcal{M}_0$ est relativement complet pour $\mathcal{S}_0$ si

$[\![\mathcal{S}_0]\!] = \mathcal{M}_0 \cap [\![\mathcal{S}]\!]$ (alors que la complétude forte serait $[\![\mathcal{S}_0]\!] = \mathcal{M}_0$). On peut noter qu'un résultat de complétude relative interdit les modèles de $\mathcal{S}$ vus comme modèles de $\mathcal{S}_0$. Dès que $\mathcal{S}_0$ est une restriction non triviale de $\mathcal{S}$ (en particulier $[\![\mathcal{S}_0]\!] \neq [\![\mathcal{S}]\!]$), $\mathcal{M}$ n'est pas un modèle relativement complet de $\mathcal{S}_0$, puisque sinon $[\![\mathcal{S}_0]\!] = \mathcal{M} \cap [\![\mathcal{S}]\!] = [\![\mathcal{S}]\!]$. Un modèle relativement complet donne une caractérisation sémantique des preuves de $\mathcal{S}$ qui sont dans $\mathcal{S}_0$. Grâce aux techniques développées dans [TdF03] sur les expériences obsessionnelles, nous avons pu montrer la complétude relative de nos modèles de ELL et SLL.

On peut noter qu'afin d'interpréter des systèmes extensionnellement complets pour les temps élémentaire et polynomial, nous devons aller au-delà des fragments propositionnels. La solution usuelle consiste à prendre en compte la quantification du second ordre, ce qui complique beaucoup la construction et l'étude des modèles. Nous avons préféré utiliser des versions non typées des systèmes ELL et SLL, ce qui garantit également la complétude extensionnelle. Pour les interpréter, on introduit dans notre modèle un ensemble D « à la Scott » tel que $D \times D$ et $\mathcal{M}_{fin}(D)$ s'injectent dans D.

Une question naturelle qui se pose est de chercher à traiter LLL de la même manière afin d'obtenir en particulier un modèle relativement complet (le modèle stratifié l'est pour ELL, mais sa version stratifiée bornée pour LLL ne l'est pas). Ce problème reste ouvert.

7 Mesure sémantique du temps de calcul en logique linéaire

L'approche précédente consistait à chercher des modèles spécifiquement adaptés à un système allégé particulier (dont on connaît la classe de complexité associée, par correction et complétude extensionnelle). Une alternative possible est de chercher à calculer la complexité (temps de calcul notamment) de n'importe quelle preuve de LL, de manière à déterminer ensuite si une preuve donnée représente un prédicat de complexité polynomiale par exemple. Un intérêt majeur de cette approche est de se libérer des problèmes d'expressivité intentionnelle puisqu'il n'y a plus de restriction *a priori* sur l'espace des algorithmes autorisés.

Un obstacle immédiat apparaît si on cherche une approche sémantique à cette question. En effet, un modèle dénotationnel fournit un invariant du calcul. L'interprétation d'une preuve et de sa forme normale est la même. Par conséquent il n'est pas possible d'extraire une information sur le temps de normalisation à partir de l'interprétation dénotationnelle d'une preuve. Une première manière de contourner ce point consiste à définir une paire d'interprétations : la première définissant un modèle dénotationnel, la seconde n'étant pas invariante par réduction mais la plus proche possible de la première en ajoutant juste l'information nécessaire pour évaluer le temps de normalisation. C'est ce que nous allons présenter à l'aide d'un modèle de jeux [DLL08]. Une second solution consiste, non pas à déterminer (à partir de sa sémantique) le temps de normalisation d'*une* preuve mais, étant données *deux* preuves normales de $A \vdash B$ et $B \vdash C$, à déterminer le temps de normalisation de la preuve obtenue par coupure sur B, grâce aux interprétations des deux preuves d'origine. C'est ce qui est proposé dans [dCPTdF09] à l'aide du modèle relationnel.

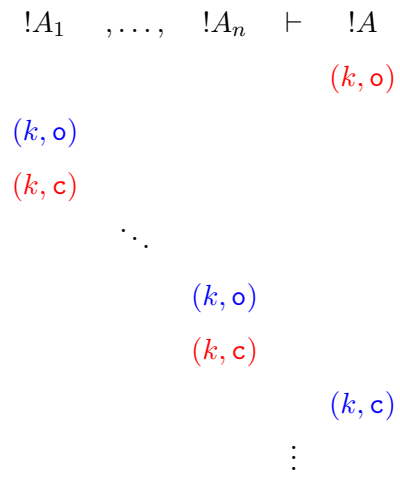
Du fait de leur proximité avec la syntaxe, les modèles de jeux sont un candidat naturel pour étudier précisément la normalisation des preuves. Les principaux modèles de jeux de la logique linéaire se concentrent sur le fragment intuitionniste IMELL. Se restreindre à ce fragment n'est pas trop problématique pour nous puisqu'il reste suffisamment expressif pour représenter tout ce que l'on peut faire en λ-calcul typé. Ceci étant, un défaut majeur des modèles de jeux pour ce qui nous intéresse ici est leur caractère affine : ils sont incapables d'inspecter les parties de preuves/programmes qui seront effacées pendant la normalisation puisque ces parties de preuve ne participent pas du tout à l'interprétation dans les jeux. Par conséquent il n'est pas possible

de mesurer dans les jeux le temps nécessaire à effacer un morceau de preuve (qu'il soit gros ou non) alors que ceci apparaît explicitement dans la syntaxe. De même, du calcul (aussi long soit-il) effectué dans un morceau de preuve, que l'on efface ensuite dans la syntaxe, est totalement invisible dans les jeux.

On s'intéresse à un modèle de jeux construit à partir des constructions multiplicatives usuelles sur les jeux à parties (par exemple [Abr97]) ce qui fournit une catégorie monoïdale symétrique fermée et donc permet d'interpréter IMLL.

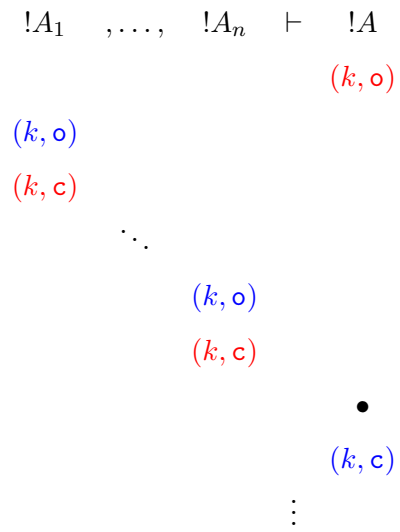
Définition (Jeu)
Un *jeu* est un triplet $(M_A, \lambda_A, \mathcal{P}_A)$ formé d'un ensemble de coups M_A, d'une fonction de polarité $\lambda_A : M_A \to \{O, J\}$ et d'un ensemble de parties $\mathcal{P}_A$ qui sont des suites de coups de polarités alternées et commençant par un coup Opposant.

Pour interpréter IMELL, on définit habituellement une construction de jeux $\sharp A$ en créant des copies de A indicées par des entiers : $M_{\sharp A} = \mathbb{N} \times M_A$, $\lambda_{\sharp A}(k, \mathtt{m}) = \lambda_A(\mathtt{m})$ et $\mathtt{s} \in \mathcal{P}_{\sharp A}$ si, pour tout k, $\mathtt{s} \restriction_k \in \mathcal{P}_A$ (où $\mathtt{s} \restriction_k$ est obtenue en ne gardant que les coups d'indice k dans $\mathtt{s}$). Utiliser directement cette construction pour interpréter ! pose les problèmes de caractère affine que nous mentionnions. Afin de les éviter, on introduit une construction de *décalage* ajoutant deux coups en début de parties : $\updownarrow A = (M_A \cup \{\mathtt{o}, \mathtt{c}\}, \lambda_A \cup \{(\mathtt{o}, O), (\mathtt{c}, J)\}, \{\varepsilon, \mathtt{o}\} \cup \mathtt{o}\mathtt{c}.\mathcal{P}_A)$. Ceci permet d'ajouter du « temps » en début de partie et on définit $!A = \sharp\updownarrow A$. Le défaut de cette construction est d'introduire de la séquentialité dans l'interprétation des règles comme le montre le cas de la règle de promotion multi-fonctorielle :



On définit ainsi une interprétation de IMELL [DLL08] : la stratégie associée à une preuve est, à l'ordre des coups $\mathtt{o}$ et $\mathtt{c}$ par rapport aux autres près, invariante par élimination des coupures. Il convient alors d'enrichir ce modèle avec des informations qui ne sont pas invariantes par réduction (même modulo permutation de coups) mais permettent de mesurer la complexité. Lors de la construction $\updownarrow A$, on introduit des jetons $\bullet$ (dans l'esprit de [Ghi05]) entre les coups $\mathtt{o}$ et $\mathtt{c}$. Ces jetons ne se comportent pas exactement comme des coups puisqu'ils n'ont pas de polarité et s'accumulent lors de la composition : alors que les coups dans B disparaissent lors d'une composition de $\sigma : A \to B$ avec $\tau : B \to C$, les jetons présents dans B sont conservés.

Les jetons sont introduits à travers les règles de promotion multi-fonctorielle :



Le point important est qu'il est possible de définir une mesure de complexité à l'intérieur de ce modèle de jeux avec jetons. Si $\top$ est le jeu vide (sans coup), les parties maximales du jeu $\updownarrow\top$ sont de la forme $\mathsf{o} \bullet \cdots \bullet \mathsf{c}$, et sont donc en bijection avec $\mathbb{N}$ (en comptant le nombre de jetons). On définit pour tout jeu A, une stratégie $\mathsf{TA}_A : A \to \updownarrow\top$ afin d'analyser le temps de normalisation des preuves : si π est une preuve de A, et si $[\![\pi]\!] : A$ est la stratégie avec jetons associée, TA_A définit une fonction $\mathcal{C}$ des preuves de IMELL dans $\mathbb{N}$ en composant $[\![\pi]\!]$ avec TA_A et en extrayant l'entier obtenu dans $\updownarrow\top$. Le calcul de la complexité associée à une stratégie est ainsi internalisé dans le modèle de jeux via TA, de plus on peut noter que TA_A ne dépend que de A.

Pour simplifier la définition de TA, on se restreint à la *réduction de surface*, c'est-à-dire ne réduisant pas de coupure à l'intérieur des boîtes dans les réseaux. Cette stratégie de réduction est suffisamment puissante pour calculer la forme normale d'un réseau représentant un entier par exemple.

Théorème (Adéquation de la mesure de complexité)
La longueur maximale d'une réduction de surface de π (à laquelle on ajoute la taille du plus grand réduit de π) et la mesure $\mathcal{C}(\pi)$ (à laquelle on ajoute la taille de π) sont polynomialement reliées.

L'objectif est désormais d'unifier les deux approches de construction de modèles que nous avons présentées, en trouvant des modèles à complexité intrinsèquement bornée. C'est-à-dire des modèles qui non seulement permettent d'interpréter une (ou des) logique(s) allégée(s) (ce qui offre une garantie d'expressivité) mais assurent également une borne de complexité sur les preuves ou programmes interprétables dans le modèle. Ce type d'approche apparaît dans [DLH05]. Il nous parait naturel d'essayer de travailler dans le contexte de la sémantique des jeux [Abr02, Red07].

Troisième partie

Logique linéaire différentielle et concurrence

[EL07] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. In Luis Caires and Vasco T. Vasconcelos, editors, *Eighteenth International Conference on Concurrency (CONCUR)*, volume 4703 of *Lecture Notes in Computer Science*, pages 333–348. Springer, September 2007.

[HL08] Kohei Honda and Olivier Laurent. An exact correspondence between a typed pi-calculus and polarised proof-nets. Submitted to *Theoretical Computer Science*, November 2008.

[EL09] Thomas Ehrhard and Olivier Laurent. Acyclic solos and differential interaction nets. Submitted to *Logical Methods in Computer Science*, 2009.

L'étude de la correspondance de Curry-Howard est quasi-intégralement consacrée au cas du calcul séquentiel alors qu'une part très importante de l'informatique concerne la programmation concurrente et les systèmes distribués et mobiles, ce qui se traduit par d'importants développements autour des algèbres de processus concurrents. Il n'y a que très peu de passerelles avec la logique, et celles-ci sont toujours obtenues finalement en appliquant une modification ad hoc des systèmes considérés. Une raison à cela est le manque de compréhension des aspects non déterministes de l'élimination des coupures en logique. De ce point de vue, l'introduction de la logique linéaire différentielle a profondément modifié le paysage : l'élimination des coupures est déterministe mais une preuve est constituée d'une superposition (combinaison linéaire) de preuves simples. Il s'agit de la traduction syntaxique de modèles de la logique linéaire construits à partir d'espaces vectoriels. Il semble alors possible d'interpréter la réduction d'une preuve π en une somme $\pi_1 + \pi_2$ comme la capacité du processus représenté par π à se réduire de manière non déterministe en π_1 ou en π_2.

Dans un premier temps nous présentons comment une extension (apparemment) tout à fait ad hoc des réseaux de preuve polarisés permet d'établir des liens très étroits avec une version typée du π-calcul. Grâce aux réseaux d'interaction différentiels, on établit ensuite une autre connexion entre π-calcul et logique linéaire, mais cette fois sans modifier de manière arbitraire le système logique utilisé. On peut enfin vérifier que l'analyse par la logique linéaire différentielle permet de justifier logiquement l'extension utilisée précédemment en logique linéaire polarisée.

On aboutit de cette manière à des traductions de calculs de processus dans la logique linéaire différentielle, sans qu'il soit nécessaire de modifier celle-ci. Vues comme les premières pierres d'une correspondance de Curry-Howard concurrente, elles permettent d'envisager sérieusement des transferts de technologies à partir des outils développés en logique linéaire différentielle. Les travaux existant par ailleurs nécessitaient de quitter le monde de la logique et par conséquent hypothéquaient sérieusement les chances de ré-utiliser les outils développés auparavant en théorie de la démonstration.

8 π-calcul typé et réseaux polarisés

Partant d'un π-calcul asynchrone, M. Berger, K. Honda et N. Yoshida ont construit une restriction du π-calcul séquentielle et déterministe qui permet de coder exactement le $\lambda\mu$-calcul [HYB04]. Par ailleurs, du fait des liens étroits entre logique classique constructive, $\lambda\mu$-calcul et logique linéaire polarisée, les *réseaux de preuve polarisés* fournissent une syntaxe graphique pour le $\lambda\mu$-calcul [Lau03]. Via ce point commun qu'est le $\lambda\mu$-calcul, on trouve ainsi une connexion entre π-calcul et réseaux polarisés. Lorsque l'on cherche à comprendre plus précisément ce lien, un premier élément est que l'on obtient des traductions directes entre ces deux systèmes qui s'abstraient totalement du $\lambda\mu$-calcul et sont de nature plus primitive. Ainsi même si le $\lambda\mu$-calcul a pu servir de point de contact initial, il apparaît que π-calcul et réseaux polarisés sont beaucoup plus proches l'un de l'autre que du $\lambda\mu$-calcul. Un second élément est que si les réseaux polarisés fournissent ainsi une syntaxe graphique pour un π-calcul restreint, on peut essayer de relâcher les contraintes qui ont été imposées aux π-termes et de comprendre leur signification en termes de réseaux.

On considère un π-calcul polyadique construit de la manière suivante :

$$P \quad ::= \quad !x(\vec{y}).P \quad | \quad \bar{x}(\vec{y})P \quad | \quad P|P \quad | \quad \nu x P \quad | \quad 1$$

où $!x(\vec{y}).P$ est une réception répliquée (avec $\vec{y}$ lié dans P), $\bar{x}(\vec{y})P$ est une émission asynchrone privée (avec $\vec{y}$ lié dans P) qui peut s'écrire $\nu\vec{y}(\bar{x}\langle\vec{y}\rangle|P)$ en π-calcul usuel, $P|Q$ est la mise en parallèle de deux processus, $\nu x P$ est la restriction (x est lié dans P) et 1 est le processus inactif.

La règle de réduction cruciale est celle de communication entre une émission et une réception :

$$\bar{x}(\vec{y})P \mid !x(\vec{y}).Q \quad \rightarrow \quad \nu\vec{y}(P \mid Q) \mid !x(\vec{y}).Q$$

Le système de typage présenté dans [HYB04] peut être décomposé de la manière suivante :

1. *Localité* : les canaux reçus lors d'une réception sont nécessairement utilisés en émission.
2. *Déterminisme en réception* : un canal ne peut pas être utilisé simultanément (*i.e.* en parallèle) pour deux réceptions.
3. *Déterminisme en émission* : un processus contient exactement une unique émission active.
4. *Acyclicité* : la séquentialité induite par les préfixes de réception répliquée définit une relation sur les canaux qui est acyclique.

Nous avons montré, avec Kohei Honda [HL08], que les π-termes vérifiant toutes ces contraintes correspondent à des réseaux de preuve polarisés (satisfaisant le critère de correction). Plus précisément, il est possible de relâcher les contraintes de typage sur le π-calcul et d'identifier les conditions associées dans les réseaux. Le résultat est étonnamment simple : l'acyclicité du π-calcul correspond à l'acyclicité orientée du critère de correction polarisé, le déterminisme en émission correspond à la connexité du critère de correction, le déterminisme en réception correspond à l'unicité de la prémisse dans la règle de promotion (contrainte que la logique linéaire différentielle nous autorisera à relâcher grâce au principe de co-contraction), enfin la localité correspond à la contrainte de contexte ?Γ de la règle de promotion (nous ne toucherons pas à cette dernière hypothèse faute de quoi la logique linéaire perd rapidement son sens). Ceci permet d'éclairer les conditions de typage du π-calcul sous un nouveau jour, un peu comme les conditions de sémantique des jeux (parenthésage, rigidité, totalité, ...) apportent un nouveau regard sur les constructions syntaxiques des langages fonctionnels.

Ainsi en ne gardant que la condition de localité du système de typage du π-calcul, on obtient une équivalence avec un système de structures de preuve polarisées (réseaux polarisés sans critère de correction) dans lesquelles les nœuds ! peuvent avoir plusieurs prémisses (voir ci-dessous).

Cette extension (*a priori* ad hoc : nous verrons un peu plus loin comment la logique linéaire différentielle la justifie) de la logique linéaire polarisée avec plusieurs boîtes associées à un même nœud ! permet d'introduire du non déterminisme dans l'élimination des coupures. Si une contraction duplique toutes les boîtes et un affaiblissement les efface toutes, une déréliction sélectionne l'une d'entre elles de manière non déterministe, l'ouvre et supprime les autres (on trouvera un exemple dans la section 9.5).

En tenant compte de la polarisation et afin d'optimiser la représentation des π-termes par les réseaux, on fait des choix spécifiques dans la syntaxe des réseaux. On utilise quatre types de nœuds différents : $\otimes$, $\parr$, ? et !.

$$\frac{!N_1 \quad \cdots \quad !N_n}{\bigotimes_{1\leq i\leq n} !N_i}\ n \geq 0 \qquad \frac{?P_1 \quad \cdots \quad ?P_n}{\parr_{1\leq i\leq n} ?P_i}\ n \geq 0$$

$$\frac{P \quad \cdots \quad P}{?P}\ n \geq 0 \text{ prémisses} \qquad \frac{N \quad \cdots \quad N}{!N}\ n > 0 \text{ prémisses}$$

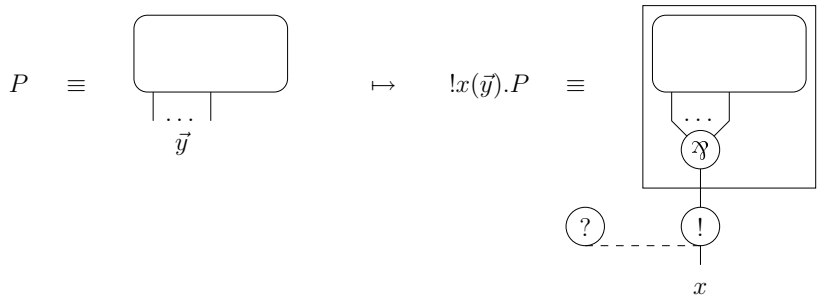
Dans ce contexte précis, une structure de preuve est une forêt formée à partir des nœuds ci-dessus. Les racines de la forêt sont les conclusions de la structure de preuve et les feuilles sont obtenues à l'aide de nœuds $\otimes$, $\parr$ ou ? d'arité 0. L'utilisation des nœuds $\otimes$ et $\parr$ d'arité arbitraire (et la contrainte de typage imposant que deux nœuds $\otimes$ ou deux nœuds $\parr$ ne puissent pas être immédiatement l'un au-dessus de l'autre) est justifiée par l'associativité de ces connecteurs ou encore par la propriété de focalisation. À chaque prémisse de nœud ! est associée une boîte

contenant une sous-structure de preuve. Enfin on traite les coupures de manière spécifique pour coller à l'idée que l'interaction provient du fait que deux agents veulent effectuer des actions duales sur une même adresse [Gir01]. Il n'y a donc pas de nœuds coupures mais des *arêtes de coupure* entre la conclusion d'un nœud ? et celle d'un nœud !.

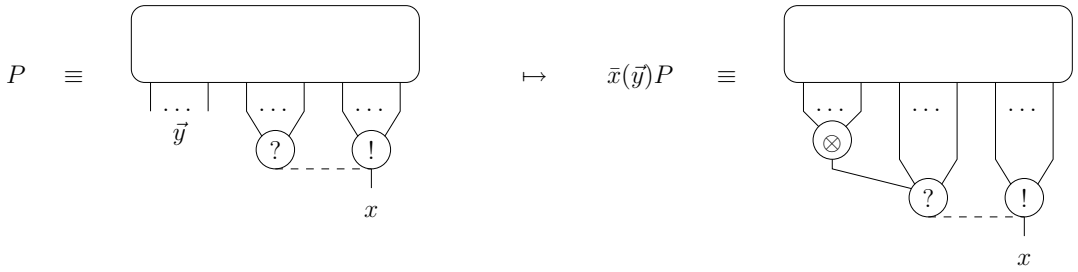
La traduction d'un π-terme est basée, pour l'essentiel, sur l'idée d'associer, à chaque nom libre x, une conclusion ? et possiblement une conclusion ! (uniquement si le π-terme contient des réceptions sur x) dans le réseau correspondant. Des arêtes de coupure sont introduites entre la conclusion d'un ? et celle d'un ! lorsqu'ils correspondent au même nom de canal.

Ainsi on a, construction par construction :

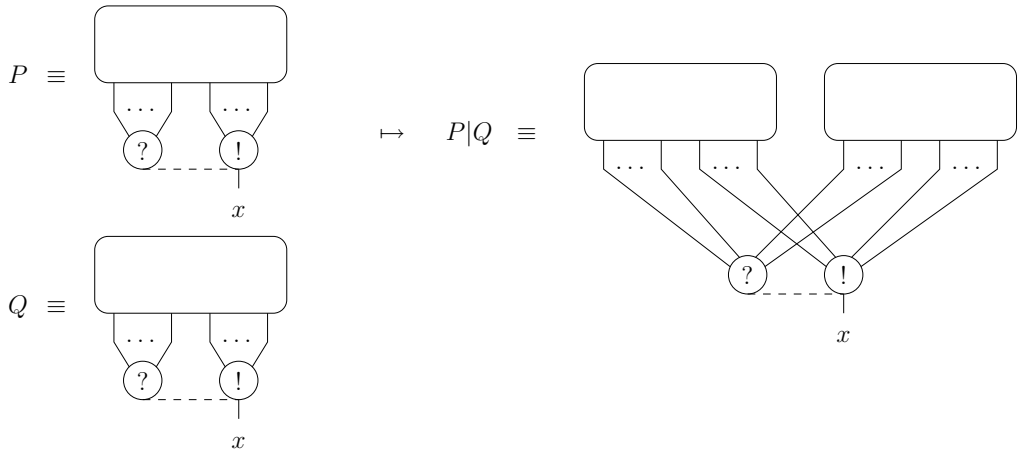
- La traduction du π-terme 1 est le réseau vide.
- La traduction de $!x(\vec{y}).P$ est obtenue en ajoutant un nœud $\parr$ dont les prémisses sont les nœuds ? associés aux $\vec{y}$. On ajoute également un nœud ! avec le $\parr$ comme prémisse et tout le réseau (excepté le !) comme boîte de cette prémisse. On complète avec une arête de coupure entre le nœud ! et un nœud ?.



- La traduction de $\bar{x}(\vec{y})P$ est obtenue en ajoutant un nœud $\otimes$ dont les prémisses sont les nœuds ! associés aux $\vec{y}$. Si x est libre dans P, la conclusion du $\otimes$ est ajoutée comme prémisse du ? associé à x, sinon on ajoute un nouveau nœud ? sous le $\otimes$.

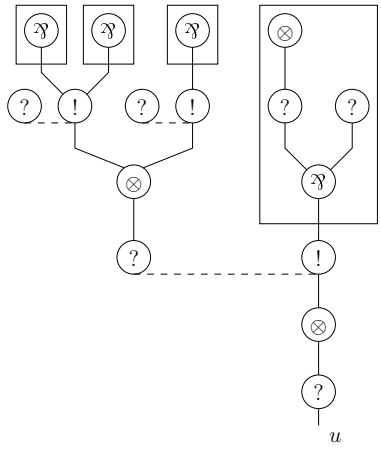


- La traduction de $P|Q$ est obtenue en identifiant les nœuds ? et ! des traductions de P et Q qui correspondent aux mêmes noms. On augmente ainsi l'arité de ces nœuds.



– La traduction de $\nu x P$ est la même que celle de P.

La traduction du π-terme $\bar{u}(x)\,(\,\bar{x}(yz)(!y.1\,|\,!y.1\,|\,!z.1)\,|\,!x(yz).\bar{y}1\,)$ est ainsi :



Théorème (Simulations)
Deux π-termes correspondent à la même structure de preuve si et seulement si ils sont structurellement congruents.

La réduction des structures de preuve simule celle des π-termes et la réduction des π-termes simule celle des structures de preuve.

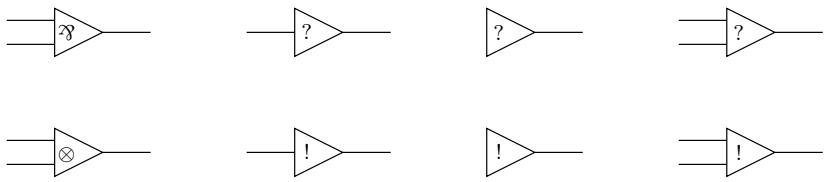
Ce résultat s'applique naturellement au cas particulier des π-termes typés à la [HYB04] et des réseaux de preuve corrects.

9 Réseaux d'interaction différentiels

Un progrès majeur par rapport aux travaux précédents, notamment [HL08], a été obtenu grâce à l'utilisation de la logique linéaire différentielle [ER06] qui permet une justification logique des constructions.

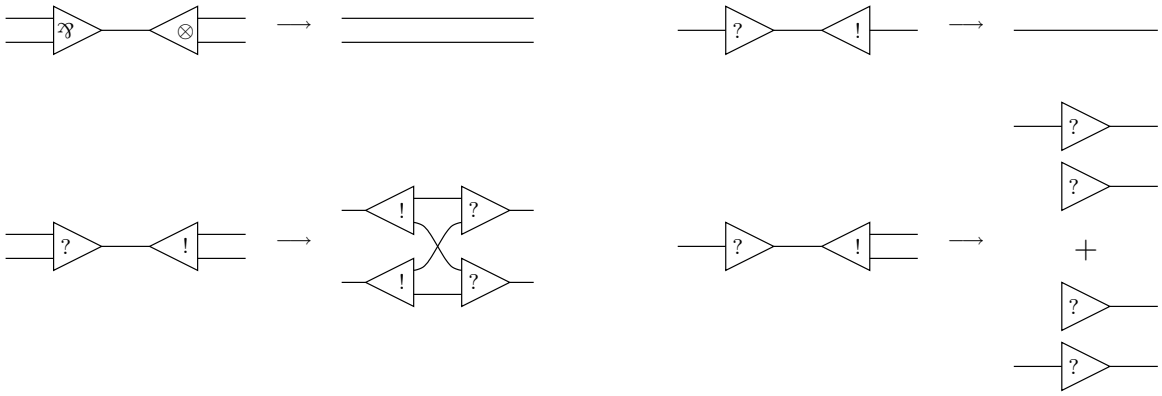
Cette extension de la logique linéaire provient de travaux sémantiques sur des modèles d'espaces vectoriels pour LL et introduit dans la syntaxe des opérations correspondant au calcul différentiel qui augmentent la symétrie entre les connecteurs exponentiels ? et !. Les principes de co-déréliction $A \multimap !A$, de co-contraction $!A \otimes !A \multimap !A$ et de co-affaiblissement $1 \multimap !A$ deviennent valides. Cette logique est dotée d'une opération de somme entre preuves qui va permettre de représenter les aspects non déterministes du calcul : π se réduit en $\pi_1 + \pi_2$ revient à dire que π peut se réduire de manière non déterministe soit en π_1 soit en π_2.

Les réseaux différentiels que nous utilisons ici sont basés sur les cellules suivantes :



Il s'agit des cellules par, déréliction, affaiblissement, contraction, tenseur, co-déréliction, co-affaiblissement, et co-contraction. Plus généralement, on utilisera des cellules ⅋ et ⊗ d'arités quelconques.

Les principales étapes de réduction (voir [EL07] pour une présentation complète) que nous utiliserons sont les suivantes :



Les deux premières correspondent à du simple « routage ». La troisième est une forme forte de duplication qui est cruciale dans le comportement des zones de communication que nous allons définir. La dernière enfin incarne le non déterminisme des réseaux différentiels : on interprétera la somme obtenue comme la possibilité pour le réseau d'origine de se réduire arbitrairement en l'un ou en l'autre des termes de la somme.

En remarquant que l'extension des réseaux polarisés définie dans [HL08] correspond à la co-contraction définie en logique linéaire différentielle (voir section 9.5), nous avons pu avec Thomas Ehrhard [EL07] définir une correspondance de Curry-Howard entre un système logique (non modifié spécifiquement et donc conservant ses propriétés et ses modèles dénotationnels) et un calcul de processus (le fragment finitaire du π-calcul, *i.e.* sans réplication ni récursion pour l'instant). On utilise un système de typage polarisé peu contraignant pour les réseaux différentiels, mais induisant une orientation sur les arêtes et c'est tout ce qui sera utile pour nous ici concernant ces types. L'interprétation d'un processus du π-calcul par un réseau différentiel est alors basée sur la représentation de chaque canal de communication par deux arêtes d'orientations opposées, l'une permettant de représenter les émissions effectuées par le processus sur ce canal et l'autre pour les réceptions.

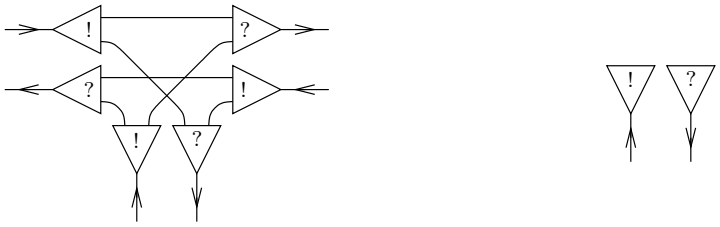
Après avoir défini les principaux ingrédients qui nous seront nécessaires pour interpréter des processus concurrents à l'aide des réseaux différentiels, nous montrerons à quelles primitives calculatoires ils correspondent. On commencera par la communication proprement dite, à travers le passage de noms en calcul des solos. On ajoutera dans un deuxième temps la séquentialité nécessaire à représenter les processus de CCS (donc sans passage de noms). On regroupera alors tous ces ingrédients pour pouvoir aborder le π-calcul. De manière à comparer l'approche décrite dans la section précédente à base de réseaux polarisés avec celle utilisée ici, on décrira finalement les principaux éléments d'une traduction du π-calcul asynchrone de Berger-Honda-Yoshida dans les réseaux différentiels.

On se restreindra à des systèmes finitaires (sans définition récursive ni réplication), sans opérateur de choix et sans test d'égalité.

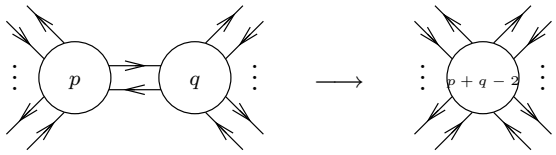
9.1 Ingrédients clefs

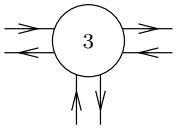
On construit une boîte à outils de réseaux différentiels qui permettront ensuite de représenter différentes primitives de communication concurrente [EL07]. On décrit à chaque fois la construction à partir des cellules des réseaux différentiels, les propriétés induites en terme de réduction que nous utiliserons par la suite, et enfin des macro-notations pour simplifier les dessins.

Zones de communication. L'ingrédient essentiel permettant à deux processus de communiquer est la *zone de communication*. Le cas central est celui de la zone ternaire (*i.e.* ayant trois arêtes entrantes et trois arêtes sortantes, correspondant à trois canaux de communication), et on utilise également souvent le cas unaire :



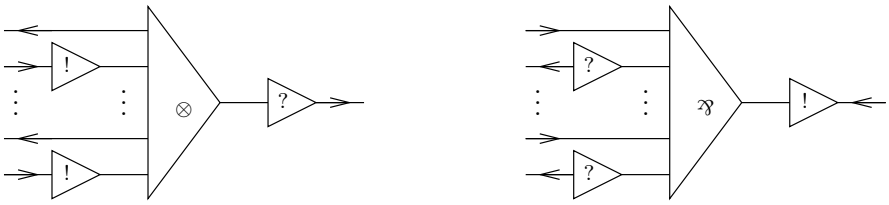
Ces zones de communication sont définies pour des arités quelconques et possèdent de bonnes propriétés d'associativité. En effet, brancher ensemble une zone d'arité p et une zone d'arité q donne par réduction une zone d'arité $p+q-2$ (chacune perd 1 d'arité du fait du branchement) :



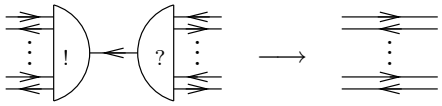
où, par exemple, 

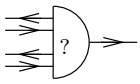
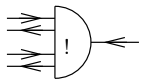
est une zone de communication d'arité 3.

Actions. Les *actions* d'émission $\bar{x}\langle\vec{y}\rangle$ et de réception $x\langle\vec{y}\rangle$ agissent comme des multiplexeurs regroupant plusieurs canaux :

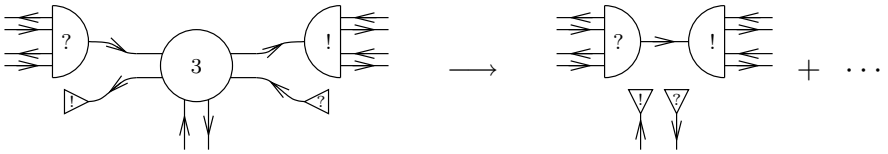


L'interaction d'une action de réception et d'une action d'émission de même arité engendre un démultiplexage :



où  et  sont des actions d'émission et de réception. Les canaux apportés en « arguments » sont ainsi connectés deux à deux.

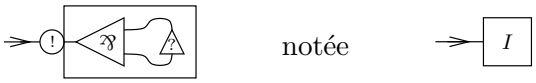
On a dit que l'on allait représenter les canaux par des paires d'arêtes d'orientations opposées, or les actions que nous venons de présenter n'ont qu'une arête « en sortie ». On associe donc souvent un affaiblissement à une action d'émission et un co-affaiblissement à une action de réception de manière à ne travailler qu'avec des paires d'arêtes. La seconde propriété importante des actions est que lorsqu'une émission (associée à un affaiblissement) et une réception (associée à un co-affaiblissement) sont connectées à la même zone de communication, elles ont la possibilité de communiquer :



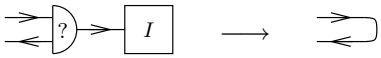
Les autres termes de la somme (dans le réduit) correspondent aux cas où les deux actions n'interagiront pas ensemble mais potentiellement avec d'autres actions du réseau.

Transistors. Puisque le calcul dans les réseaux d'interaction se fait par connexion de deux cellules par leurs ports principaux, afin d'inactiver un rédex il suffit de couper le fil reliant deux ports principaux ou, moins violemment, de connecter les deux extrémités du fil aux ports auxiliaires d'une cellule. C'est sur cette idée qu'est basée l'introduction de séquentialité dans le calcul.

On définit la « promotion d'identité » (de type $!(?\alpha^{\perp} \parr \alpha)$) :



qui permet de transformer une action d'émission (cette utilisation de l'action est totalement différente de celle présentée à la section précédente) en un simple fil :



Lorsque l'on souhaite introduire de la séquentialité, il est ainsi possible de « geler » l'interaction entre deux ports principaux de cellules en introduisant une action d'émission jouant le rôle de « transistor ». Le calcul sera ré-activé grâce à une promotion d'identité. Ci-dessous : un rédex, une version désactivée de ce rédex via l'introduction d'une action d'émission (utilisation comme transistor), une version en voie de ré-activation par branchement d'une identité promue, puis le rédex d'origine (obtenu alors par réduction).

Notations. On rappelle les macro-notations que nous avons introduites : zone de communication (d'arité 3 dans l'exemple), action d'émission, action de réception et identité promue.

Sur la base des constructions que nous venons de présenter et des propriétés induites, nous allons désormais voir comment les assembler pour représenter différents calculs concurrents.

9.2 Passage de noms et calcul des solos

L'introduction de séquentialité explicite dans le calcul complique l'interprétation dans les réseaux différentiels qui sont avant tout un modèle de calcul très local et très asynchrone. Il se trouve que ce problème a en fait déjà été abordé dans le cadre des calculs de processus : le calcul des solos [LV03] est totalement asynchrone et a malgré tout été montré suffisamment expressif pour représenter le π-calcul. Nous allons donc tout d'abord étudier la représentation des solos dans les réseaux différentiels.

Les termes du calcul des solos sont basés sur des actions sans continuation :

$$P \quad ::= \quad x[\vec{y}] \quad | \quad \bar{x}[\vec{y}] \quad | \quad P|P \quad | \quad \nu x P \quad | \quad 1$$

On considère ici une version finitaire du calcul des solos (*i.e.* sans réplication).

En l'absence de continuation, le passage de noms ne peut pas se faire par substitution dans celle-ci. Le calcul des solos est basé sur un passage de noms par unification (comme dans le calcul de fusion [PV98]) :

$$\bar{x}[\vec{y}] \mid x[\vec{z}] \mid P \quad \rightarrow \quad P\{\vec{y} \equiv \vec{z}\}$$

où $\{\vec{y} \equiv \vec{z}\}$ est un unificateur (le plus général) des $\vec{y}$ avec les $\vec{z}$ (des conditions sur la liaison des variables doivent être satisfaites pour éviter d'unifier deux noms libres).

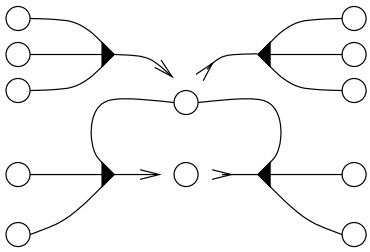
Une syntaxe graphique pour le calcul des solos existe déjà, il s'agit des diagrammes de solos [LPV01]. Ce sont des hypergraphes dans lesquels chaque nom d'un terme des solos est représenté par un nœud et chaque solo ($x[\vec{y}]$ ou $\bar{x}[\vec{y}]$) est représenté par une hyper-arête (munie d'une polarité correspondant à « réception » ou « émission ») dont les sources sont les $\vec{y}$ et la cible est x. La réduction est obtenue en identifiant les nœuds sources de deux arêtes de polarités duales et de même cible.

À l'aide du matériel présenté précédemment, il est facile d'implémenter les diagrammes de solos par des réseaux différentiels. Les nœuds sont traduits par des zones de communication d'arité appropriée, les hyper-arêtes d'émission par des actions d'émission (appariées à un co-affaiblissement) et les hyper-arêtes de réception par des actions de réception (appariées à un affaiblissement).

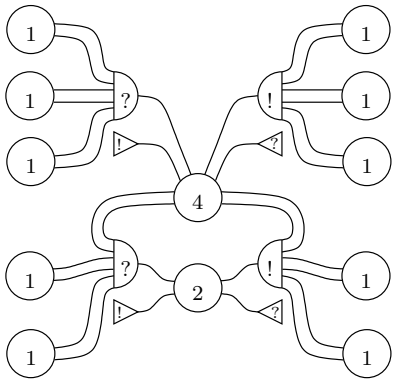
Ainsi le diagramme de solos correspondant au terme :

$$\nu uxyzy'z'abca'b'c' \left(\bar{u}[xyz] \mid u[xy'z'] \mid \bar{x}[abc] \mid x[a'b'c']\right)$$

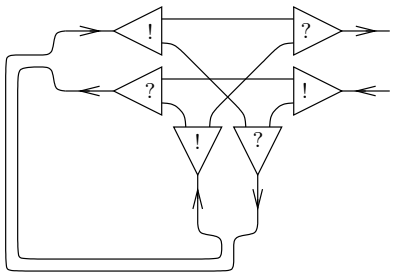
et le réseau différentiel associé sont :



et



Il reste malgré tout un problème : dans certains cas particuliers la réduction n'est pas fidèlement reproduite. Lors de la réduction $x[y] \mid \bar{x}[y] \mid P \to P$, y est identifié avec y ce qui n'a pas d'effet spécifique dans le calcul des solos (ou dans les diagrammes de solos) puisqu'on obtient la même chose avec $x[\,] \mid \bar{x}[\,] \mid P \to P$ (on a une situation similaire d'identification de y avec y dans $x[yy] \mid \bar{x}[zz] \mid P$). Par contre dans les réseaux différentiels, le fait d'essayer d'identifier un nom avec lui-même (ou d'identifier deux noms déjà égalisés par ailleurs) se traduit par une boucle sur une zone de communication :



qui ne se réduit pas en une zone de communication.

Il est malgré tout possible de caractériser une restriction du calcul des solos (le calcul des solos *acyclique* [EL09]) pour laquelle la traduction dans les réseaux différentiels simule la réduction. Cette restriction est définie par un système de typage qui brise la symétrie du passage de noms par unification du calcul des solos en identifiant certaines occurrences de noms comme associées à de l'émission ou dualement à de la réception. Le calcul obtenu satisfait quelques bonnes propriétés :

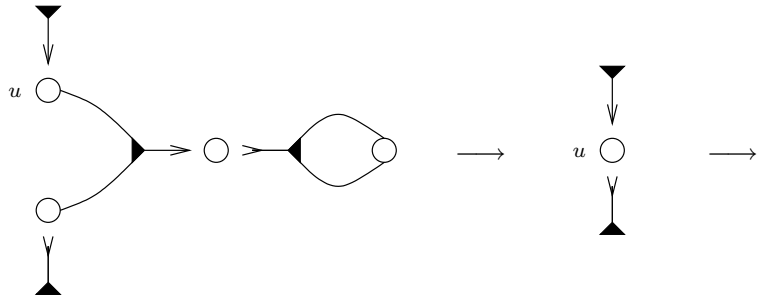
- *Préservation par réduction* : la réduction du calcul des solos préserve les types.
- *Acyclicité de l'unification* : durant la réduction un nom n'est jamais identifié à lui-même ce qui garantit un bon plongement dans les réseaux différentiels.
- *Expressivité :* l'étude du calcul des solos se justifie par l'existence d'une traduction du π-calcul [LV03] qui garantit son expressivité et on montre que l'image de cette traduction est incluse dans le calcul acyclique.

On montre ainsi en traduisant le π-calcul finitaire dans le calcul des solos acyclique puis dans les réseaux différentiels, et en définissant les systèmes de transitions appropriés (basés sur les réductions déréliction/co-déréliction pour les réseaux différentiels), qu'il existe une bisimulation entre π-calcul finitaire et réseaux différentiels (voir la fin de la section 9.4 pour quelques détails).

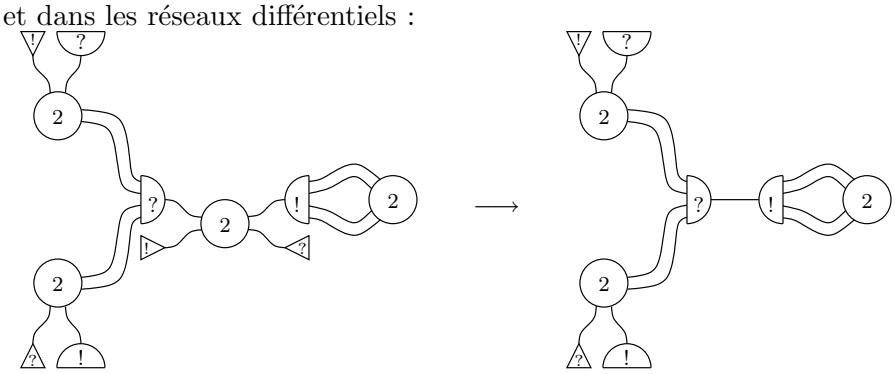
À titre d'exemple, la réduction du calcul des solos :

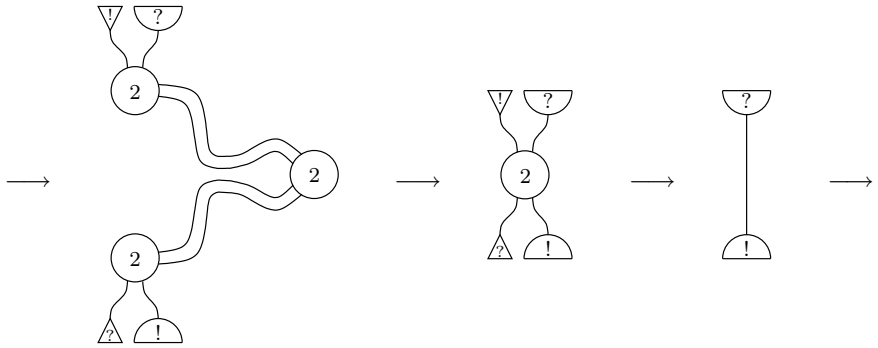
$$\nu xyz\,(\bar{x}[uy] \mid x[zz] \mid \bar{u}[\,] \mid y[\,]) \quad \to \quad \bar{u}[\,] \mid u[\,] \quad \to \quad 1$$

est simulée de la manière suivante dans les diagrammes de solos :



et dans les réseaux différentiels :





On peut constater que le solo d'émission $\bar{x}[uy]$ (et l'action d'émission correspondante dans les réseaux) joue le rôle de transistor que nous avons abordé précédemment : la communication entre l'émission $\bar{u}[\,]$ et la réception $y[\,]$ n'a pas lieu tant que ce transistor n'a pas disparu (se traduisant alors par l'identification de u et y et rendant la communication possible). C'est de là qu'est issue l'idée d'introduire les transistors pour la séquentialité.

Si nous présentons ici des calculs sans réplication, mentionnons malgré tout que la réplication est parfaitement prise en compte par le calcul des solos, les diagrammes de solos, et la traduction du π-calcul dans le calcul des solos. Obtenir des résultats aussi précis que ci-dessus, en présence de réplication, nécessitera des travaux complémentaires sur l'utilisation des boîtes dans les réseaux différentiels pour contrôler les portes auxiliaires de manière appropriée.

9.3 Séquentialité et CCS

En utilisant, comme décrit ci-dessus, les solos comme intermédiaire, on peut traduire le π-calcul finitaire dans les réseaux différentiels. Nous allons nous pencher d'un peu plus près sur cette traduction. Pour simplifier les choses, concentrons nous tout d'abord sur le cas de CCS (*i.e.* le π-calcul sans passage de noms).

Les termes de CCS finitaire (sans réplication) sont donnés par :

$$P \quad ::= \quad x.P \quad | \quad \bar{x}.P \quad | \quad P|P \quad | \quad \nu x P \quad | \quad 1$$

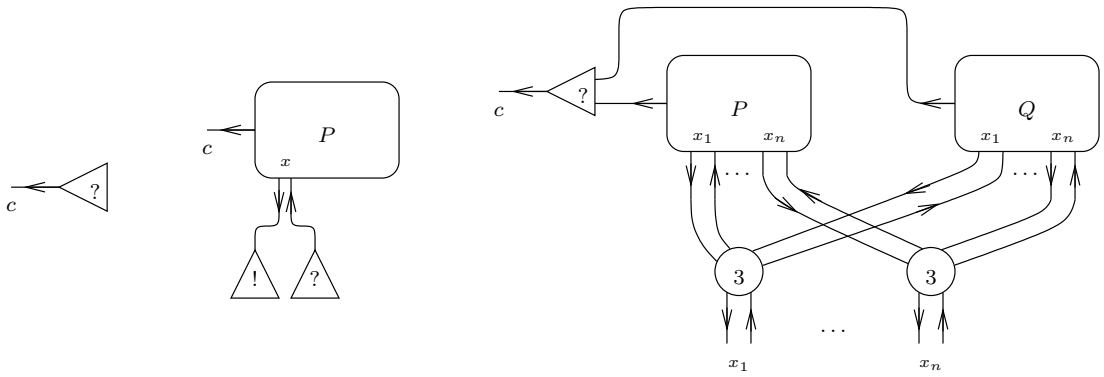
avec pour principale règle de réduction :

$$\bar{x}.P \mid x.Q \quad \rightarrow \quad P \mid Q$$

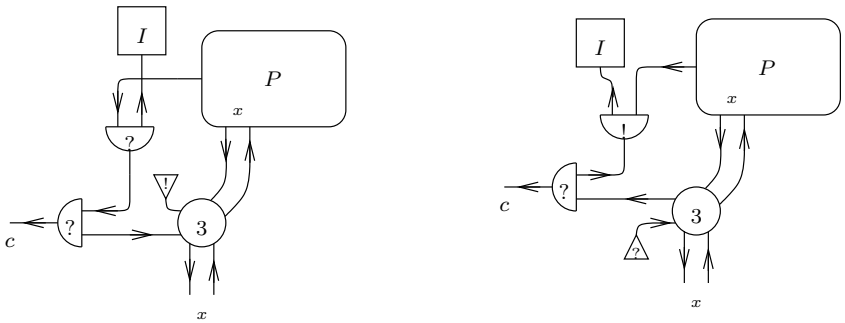
En particulier, aucune réduction ne peut se produire dans le sous-terme P de $x.P$ tant que x n'a pas interagi.

Du fait de la forte asynchronie du calcul dans les réseaux différentiels, ce type de comportement n'est *a priori* pas simple à représenter. Ayant isolé la notion de transistor grâce au passage via les solos, on peut les utiliser désormais directement pour coder la séquentialité. On peut alors traduire chacune des constructions de CCS finitaire en termes de réseaux différentiels. Des paires de fils d'orientations opposées correspondent aux noms libres du terme CCS et un fil de contrôle (utilisé pour la séquentialité) est introduit :

1 $\nu x P$ $P|Q$



$\bar{x}.P$ $x.P$

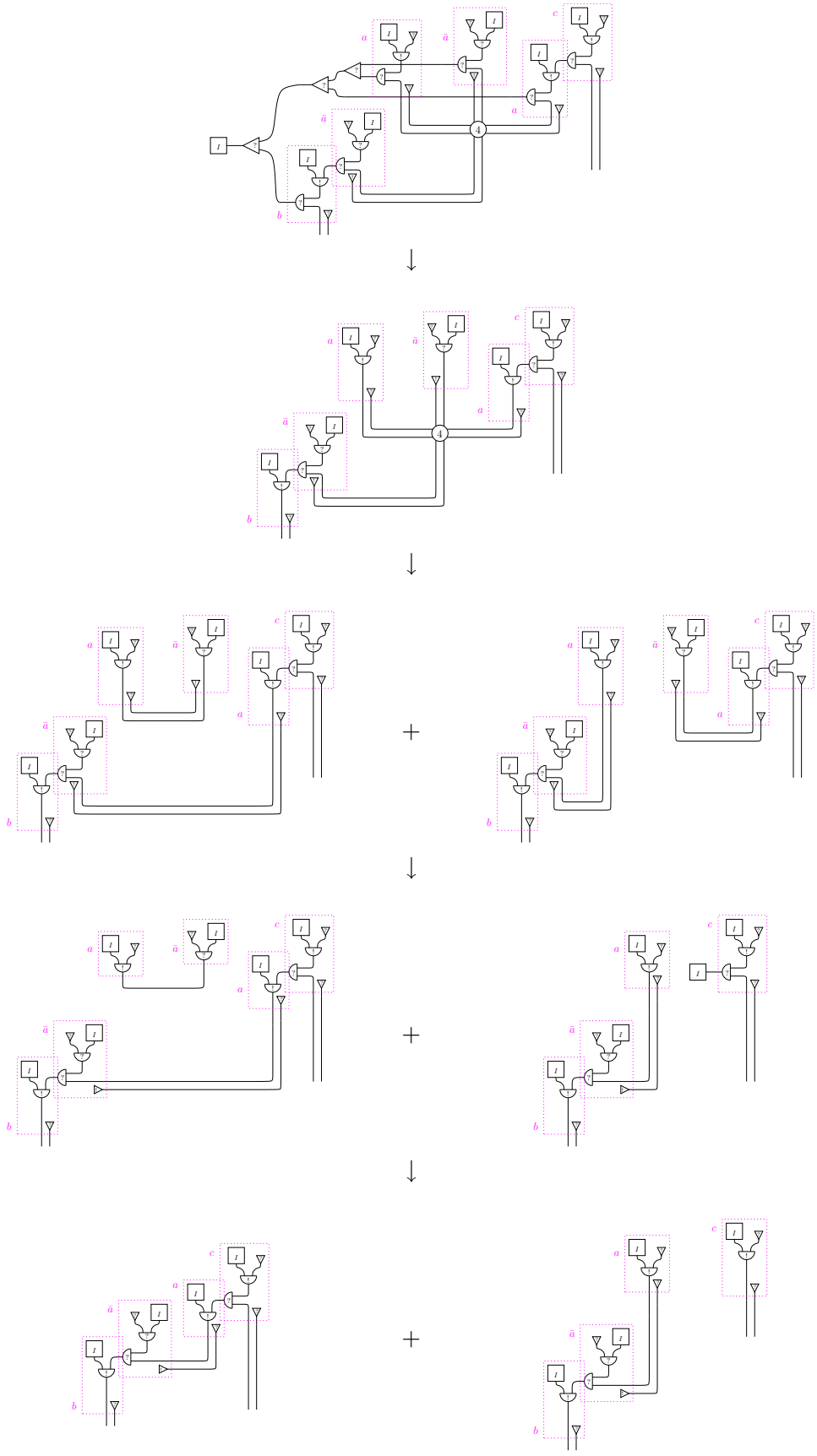


La traduction complète d'un terme de CCS finitaire est obtenue en appliquant les éléments ci-dessus puis en branchant une promotion d'identité sur le fil de contrôle.

On considère l'exemple $\nu a\,(a \mid \bar{a} \mid a.c \mid b.\bar{a})$ qui admet deux réductions :

$$\begin{aligned} \nu a\,(a \mid \bar{a} \mid a.c \mid b.\bar{a}) \quad &\rightarrow \quad \nu a\,(a.c \mid b.\bar{a}) \\ \nu a\,(a \mid \bar{a} \mid a.c \mid b.\bar{a}) \quad &\rightarrow \quad \nu a\,(a \mid c \mid b.\bar{a}) \end{aligned}$$

La réduction correspondante dans les réseaux différentiels est donnée page 48 (les rectangles pointillés ne font pas partie de la traduction, et permettent simplement de repérer les parties des réseaux associées aux différents composants du terme de CCS). Cette réduction introduit une somme correspondant aux deux choix de CCS.



9.4 π-calcul

Traduire le π-calcul revient essentiellement à ajouter le passage de noms à la traduction de CCS. En effet, les termes du π-calcul finitaire sont :

$$P \quad ::= \quad x(\vec{y}).P \quad | \quad \bar{x}\langle\vec{y}\rangle.P \quad | \quad P|P \quad | \quad \nu xP \quad | \quad 1$$

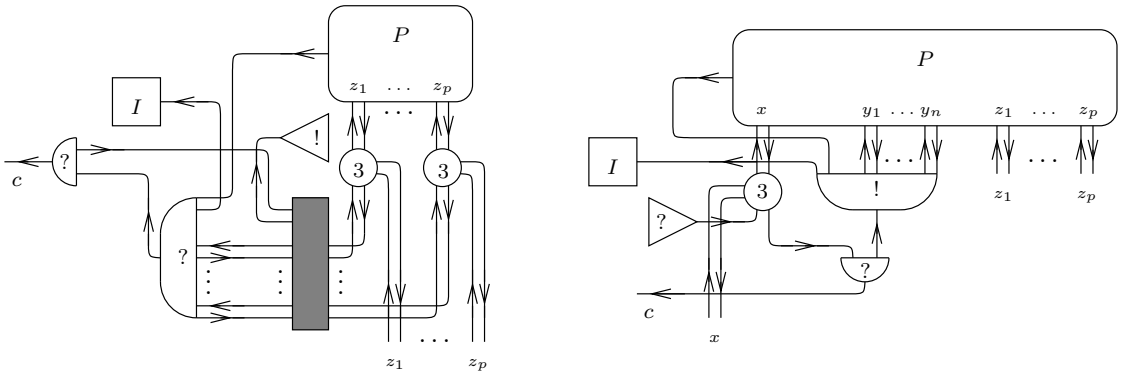
avec pour réduction :

$$\bar{x}\langle\vec{z}\rangle.P \mid x(\vec{y}).Q \quad \rightarrow \quad P \mid Q[^{\vec{z}}/_{\vec{y}}]$$

On a déjà mentionné qu'on peut obtenir une traduction en passant par le calcul des solos acyclique. On présente ici une version directe légèrement simplifiée.

On conserve la même traduction que pour CCS pour 1, νxP et $P|Q$. La traduction des préfixes d'émission et de réception devient :

$\bar{x}\langle\vec{y}\rangle.P$ $\qquad\qquad$ $x(\vec{y}).P$



où la zone grisée est constituée de zones de communication identifiant les noms égaux parmi $\{x, \vec{y}, \vec{z}\}$.

Pour relier la réduction du π-calcul finitaire et celle des réseaux différentiels, on définit des systèmes de transitions étiquetées. Contrairement à l'habitude dans l'étude des algèbres de processus, on s'intéresse aux réductions internes de façon à comparer la dynamique des deux systèmes. Pour cela on étiquette de la même manière les actions des π-termes et les (co)-dérélictions qui leurs correspondent dans le réseau différentiel associé. Une transition l/m a lieu en π-calcul lorsque qu'une réduction $\bar{x}\langle\vec{z}\rangle.P \mid x(\vec{y}).Q \rightarrow P \mid Q[^{\vec{z}}/_{\vec{y}}]$ se produit avec $\bar{x}\langle\vec{z}\rangle$ étiqueté l et $x(\vec{y})$ étiqueté m. Une transition l/m a lieu dans les réseaux différentiels lorsque que la déréliction étiquetée l interagit avec la co-déréliction étiquetée m.

Théorème (Bisimulation)
Les systèmes de transitions étiquetées associés à un π-terme finitaire et à sa traduction en réseaux différentiels sont bisimilaires.

9.5 Des réseaux polarisés aux réseaux différentiels

Afin de comparer l'approche à base de réseaux polarisés étendus et celle à base de réseaux différentiels, nous allons montrer comment étendre nos outils de réseaux différentiels pour intégrer les constructions du π-calcul de la section 8. Ceci nécessite d'utiliser des boîtes exponentielles pour représenter la réception répliquée, c'est pourquoi nous nous plaçons dans le contexte des réseaux d'interaction différentiels avec boîtes [Pag09].

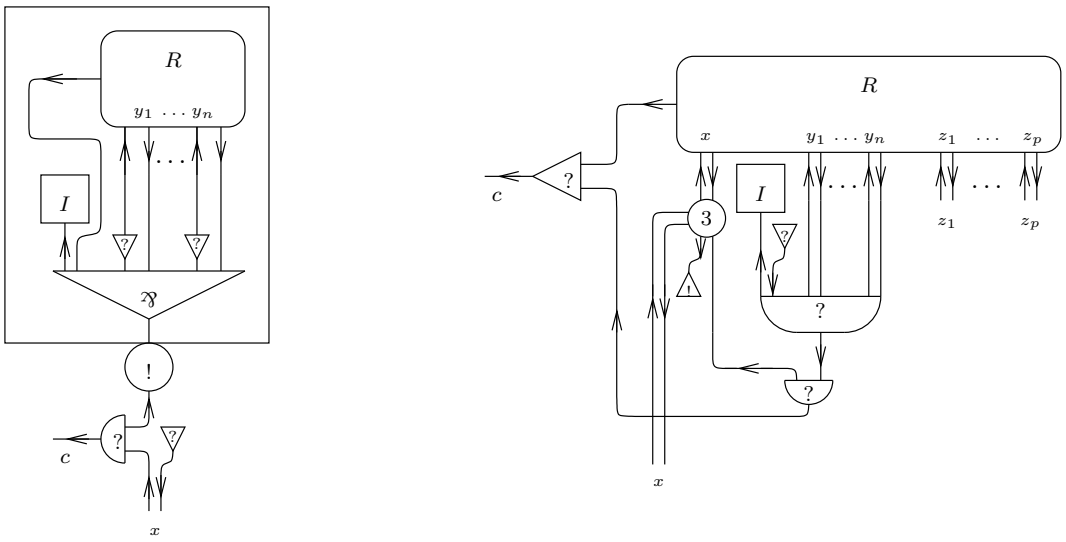
Cette section peut apparaître plus technique que les précédentes, mais ceci se justifie par le fait que les résultats présentés ici ne sont pas présents dans les articles mentionnés page 35. Il

s'agit, sans fournir tous les détails, de montrer en quoi les approches des sections 8 et 9 sont compatibles.

Pour éviter les problèmes de séquentialité au niveau des portes auxiliaires de boîtes (qui restent encore à être correctement étudiés), nous considérons ici une restriction du calcul de Berger-Honda-Yoshida : on contraint $!x(\vec{y}).R$ au cas où les seuls noms libres de R sont $\vec{y}$.

On traduit alors la réception répliquée $!x(\vec{y}).R$ et l'émission asynchrone privée $\bar{x}(\vec{y})R$ par :

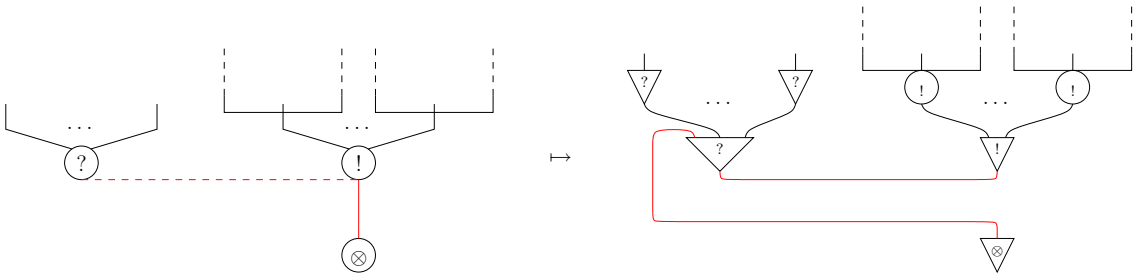
$!x(\vec{y}).R$ $\qquad\qquad$ $\bar{x}(\vec{y})R$



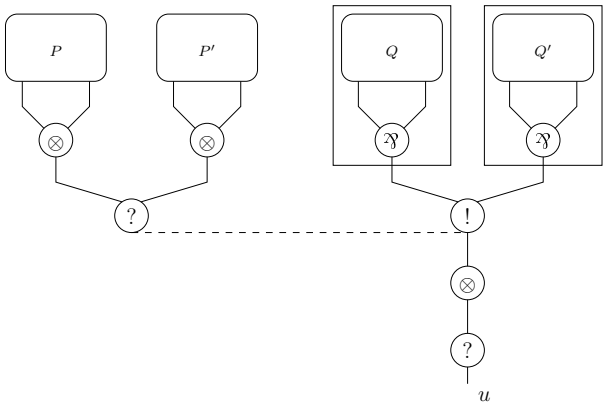
Enfin, pour mener à bien notre comparaison, nous définissons une traduction des réseaux polarisés étendus de la section 8 dans les réseaux différentiels. Les éléments clefs sont :

- les nœuds ? à plusieurs prémisses sont traduits par une contraction (ou affaiblissement) d'arité appropriée et des dérélictions ;
- les nœuds ! à plusieurs prémisses sont traduits par une co-contraction d'arité appropriée et des boîtes ;
- les arêtes de coupure entre un nœud ? et un nœud ! sont traduites en connectant la contraction obtenue à partir du nœud ? et la co-contraction obtenue à partir du nœud ! et en ajoutant une prémisse à la contraction connectée au nœud $\otimes$ situé sous le !.

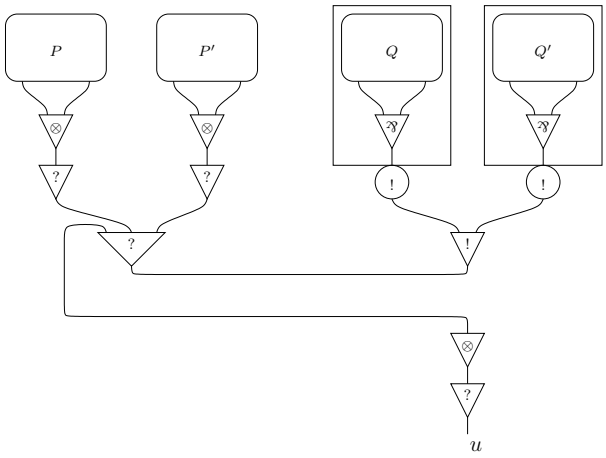
Ce qui donne :



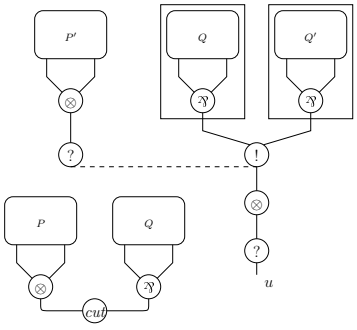
Nous allons montrer que ce codage reflète fidèlement la dynamique des réseaux polarisés dans les réseaux différentiels. En effet, le réseau polarisé :



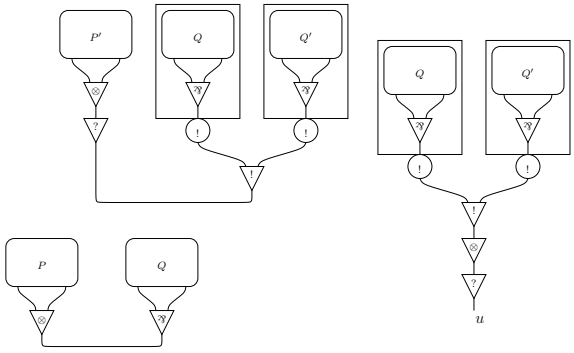
et le réseau différentiel associé :



se réduisent respectivement en :



et



où le réseau différentiel de droite est un réduit de la traduction du réseau polarisé de gauche.

Revenons au π-calcul de manière à comparer les deux traductions dans les réseaux (polarisés et différentiels). La réduction des réseaux présentée ci-dessus correspond à :

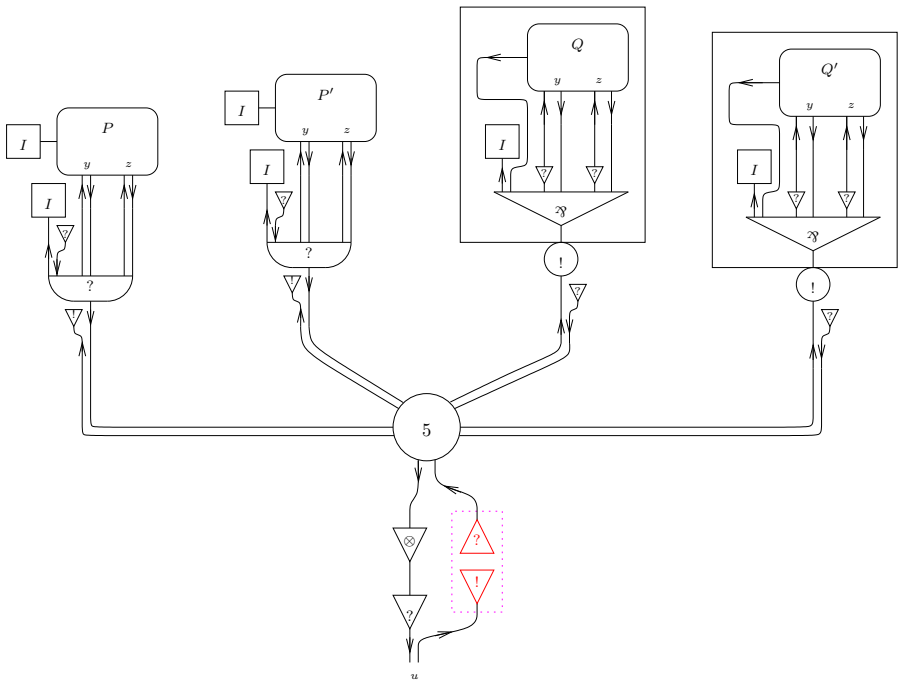
$$\bar{u}(x)\left(\bar{x}(yz)P \mid \bar{x}(yz)P' \mid !x(yz).Q \mid !x(yz).Q'\right) \\ \rightarrow \quad \bar{u}(x)\left(\nu yz(P \mid Q) \mid \bar{x}(yz)P' \mid !x(yz).Q \mid !x(yz).Q'\right)$$

via la traduction du π-calcul dans les réseaux polarisés.

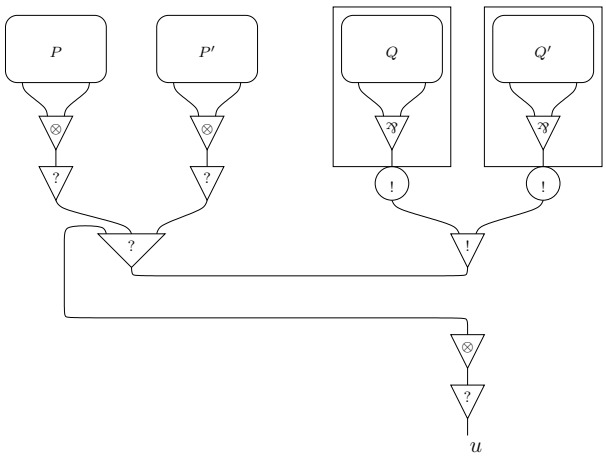
Si on regarde par ailleurs la traduction du π-terme :

$$\bar{u}(x)\left(\bar{x}(yz)P \mid \bar{x}(yz)P' \mid !x(yz).Q \mid !x(yz).Q'\right)$$

dans les réseaux différentiels, on obtient (modulo quelques réductions liées aux fils de contrôle) le réseau :



On peut montrer, en tenant compte de l'asynchronie du calcul utilisé et de la contrainte de localité (qui nous autorise, sans modifier la réduction, à substituer la partie en pointillés ci-dessus, là où on aurait dû avoir un simple fil dans le cas général), que ce réseau différentiel est calculatoirement équivalent au réseau différentiel suivant :



que l'on obtenait en passant par les réseaux polarisés étendus (*i.e.* en traduisant le π-terme en réseau polarisé puis le réseau polarisé obtenu en réseau différentiel).

Il n'est donc pas difficile d'étendre légèrement l'analyse de la concurrence par les réseaux différentiels (en intégrant $!x(\vec{y}).R$ et $\bar{x}(\vec{y})R$) comme nous venons de le présenter, pour montrer que l'extension des réseaux polarisés qui semblait totalement ad hoc est en fait logiquement valide dans un cadre différentiel : principalement le fait d'utiliser des nœuds ! à plusieurs prémisses n'est qu'une manière de représenter une co-contraction, et les arêtes de coupure une forme particulière de zone de communication.

Il convient désormais de comprendre dans quelle mesure les liens établis entre π-calcul et logique linéaire différentielle vont permettre des transferts de technologie tels que l'utilisation de modèles dénotationnels issus de la logique pour la caractérisation d'équivalences comportementales en π-calcul. De manière plus générale, un objectif consiste à faire émerger des fondements logiques pour une (des) algèbre(s) de processus.

Annexe : Logiques Linéaires

1 Logique Linéaire

1.1 Avec promotion usuelle

Formules

$$A ::= X \mid X^{\perp} \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \forall \alpha A \mid \exists \alpha A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^{\perp}}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\, \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\, \parr \qquad \frac{}{\vdash 1}\, 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\, \perp$$

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

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

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\, \forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[{}^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\, \exists$$

1.2 Avec promotion multi-fonctorielle

$$A ::= X \mid X^{\perp} \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \forall \alpha A \mid \exists \alpha A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^{\perp}}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\, \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\, \parr \qquad \frac{}{\vdash 1}\, 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\, \perp$$

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

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

$$\frac{\vdash \Gamma, ??A}{\vdash \Gamma, ?A}\,?? \qquad \frac{\vdash \Gamma, A}{\vdash ?\Gamma, !A}\,!f$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\,\forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\,\exists$$

2 Logique Linéaire Polarisée

Formules

$$P ::= X \mid P \otimes P \mid 1 \mid P \oplus P \mid 0 \mid !N \mid \exists \alpha P$$
$$N ::= X^{\perp} \mid N \parr N \mid \perp \mid N \,\&\, N \mid \top \mid ?P \mid \forall \alpha N$$

$\forall \alpha N$ et $\exists \alpha P$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash P, P^{\perp}}\,ax \qquad \frac{\vdash \Gamma, P \qquad \vdash \Delta, P^{\perp}}{\vdash \Gamma, \Delta}\,cut$$

$$\frac{\vdash \Gamma, P \qquad \vdash \Delta, Q}{\vdash \Gamma, \Delta, P \otimes Q}\,\otimes \qquad \frac{\vdash \Gamma, N, M}{\vdash \Gamma, N \parr M}\,\parr \qquad \frac{}{\vdash 1}\,1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\,\perp$$

$$\frac{\vdash \Gamma, N \qquad \vdash \Gamma, M}{\vdash \Gamma, N \,\&\, M}\,\& \qquad \frac{\vdash \Gamma, P}{\vdash \Gamma, P \oplus Q}\,\oplus_1 \qquad \frac{\vdash \Gamma, Q}{\vdash \Gamma, P \oplus Q}\,\oplus_2 \qquad \frac{}{\vdash \Gamma_{\top}, \top}\,\top$$

$$\frac{\vdash \Gamma, N, N}{\vdash \Gamma, N}\,?c \qquad \frac{\vdash \Gamma}{\vdash \Gamma, N}\,?w \qquad \frac{\vdash \Gamma, P}{\vdash \Gamma, ?P}\,?d \qquad \frac{\vdash \mathcal{N}, N}{\vdash \mathcal{N}, !N}\,!$$

$$\frac{\vdash \Gamma, N}{\vdash \Gamma, \forall \alpha N}\,\forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, P[^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha P}\,\exists$$

où $\Gamma_{\top}$ contient au plus une formule positive et $\mathcal{N}$ ne contient que des formules négatives.

3 Logique Linéaire Élémentaire

3.1 Sans $\flat$

Formules

$$A ::= X \mid X^{\perp} \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \forall \alpha A \mid \exists \alpha A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^\perp}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr \qquad \frac{}{\vdash 1}\,1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\,\perp$$

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

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

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\,\forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\,\exists$$

3.2 Avec $\flat$

Formules et pré-formules

$$A ::= X \mid X^\perp \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \forall \alpha A \mid \exists \alpha A$$
$$F ::= A \mid \flat A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^\perp}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr \qquad \frac{}{\vdash 1}\,1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\,\perp$$

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

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

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

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\,\forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\,\exists$$

4 Logique Linéaire Douce

Formules

$$A ::= X \mid X^\perp \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \forall \alpha A \mid \exists \alpha A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^\perp}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr \qquad \frac{}{\vdash 1}\, 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\perp$$

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

$$\frac{\vdash \Gamma, A, \ldots, A}{\vdash \Gamma, ?A}\, ?m \qquad \frac{\vdash \Gamma, A}{\vdash ?\Gamma, !A}\, !f$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\exists$$

5 Logique Linéaire Légère

Formules

$$A ::= X \mid X^\perp \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \S A \mid \bar{\S} A \mid \forall \alpha A \mid \exists \alpha A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^\perp}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr \qquad \frac{}{\vdash 1}\, 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\perp$$

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

$$\frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\, ?c \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\, ?w \qquad \frac{\vdash B, A}{\vdash ?B, !A}\, !f \qquad \frac{\vdash \Gamma, \Delta, A}{\vdash ?\Gamma, \bar{\S}\Delta, \S A}\, \S f$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\exists$$

6 Logique Linéaire Différentielle

Formules

$$A ::= X \mid X^\perp \mid A \otimes A \mid A \parr A \mid 1 \mid \perp \mid A \,\&\, A \mid A \oplus A \mid \top \mid 0 \mid ?A \mid !A \mid \forall \alpha A \mid \exists \alpha A$$

$\forall \alpha A$ et $\exists \alpha A$ représentent aussi bien une quantification du premier que du second ordre.

Règles

$$\frac{}{\vdash A, A^{\perp}}\, ax \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}\, cut$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\, \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\, \parr \qquad \frac{}{\vdash 1}\, 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\, \perp$$

$$\frac{\vdash \Gamma \qquad \vdash \Gamma}{\vdash \Gamma}\, + \qquad \frac{}{\vdash \Gamma}\, 0$$

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

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

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha A}\, \forall \quad x \notin \Gamma \qquad \frac{\vdash \Gamma, A[{}^{\Phi}/_{\alpha}]}{\vdash \Gamma, \exists \alpha A}\, \exists$$

Bibliographie

[Abr97] Samson Abramsky. Semantics of interaction. In Peter Dybjer and Andrew Pitts, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, pages 1–32. Cambridge University Press, 1997.

[Abr02] Samson Abramsky. Predicative copying and polynomial time. Clifford Lectures, Tulane. Available at `http://web.comlab.ox.ac.uk/oucl/work/samson.abramsky/pcpt.pdf` , 2002.

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, December 2000.

[And90] Jean-Marc Andreoli. *Proposition pour une synthèse des paradigmes de la programmation logique et de la programmation par objets*. Thèse de doctorat, Université Paris VI, June 1990.

[AR02] Andrea Asperti and Luca Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1):1–39, January 2002.

[Bai04] Patrick Baillot. Stratified coherent spaces: a denotational semantics for light linear logic. *Theoretical Computer Science*, 318(1–2):29–55, 2004.

[BE01] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Annals of Pure and Applied Logic*, 109(3):205–241, 2001.

[Ben94] Nick Benton. A mixed linear and non-linear logic: Proofs, terms and models (extended abstract). In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic*, volume 933 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 1994.

[Bie95] Gavin Bierman. What is a categorical model of Linear Logic? In Mariangiola Dezani and Gordon Plotkin, editors, *Typed Lambda Calculi and Applications '95*, volume 902 of *Lecture Notes in Computer Science*. Springer, April 1995.

[DC95] Roberto Di Cosmo. *Isomorphisms of Types*. Progress in Theoretical Computer Science. Birkhäuser, 1995.

[dCPTdF09] Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. A semantic measure of the execution time in linear logic. *Theoretical Computer Science*, 2009. To appear.

[DH01] Vincent Danos and Russell Harmer. The anatomy of innocence. In Laurent Fribourg, editor, *Computer Science Logic*, volume 2142 of *Lecture Notes in Computer Science*, pages 188–202. Springer, September 2001.

[DHR96] Vincent Danos, Hugo Herbelin, and Laurent Regnier. Game semantics & abstract machines. In *Proceedings of the eleventh annual symposium on Logic In Computer Science*, pages 394–405, New Brunswick, July 1996. IEEE, IEEE Computer Society Press.

[DJ03] Vincent Danos and Jean-Baptiste Joinet. Linear logic and elementary time. *Information and Computation*, 183(1):123–137, May 2003.

[dL07] Joachim de Lataillade. *Quantification du second ordre en sémantique des jeux — Application aux isomorphismes de types*. Thèse de doctorat, Université Paris VII, November 2007.

[DLH05] Ugo Dal Lago and Martin Hofmann. Quantitative models and implicit complexity. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, volume 3821 of *Lecture Notes in Computer Science*. Springer, 2005.

[DLL08] Ugo Dal Lago and Olivier Laurent. Quantitative game semantics for linear logic. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic*, volume 5213 of *Lecture Notes in Computer Science*, pages 230–245. Springer, September 2008.

[EL07] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. In Luis Caires and Vasco T. Vasconcelos, editors, *Eighteenth International Conference on Concurrency (CONCUR)*, volume 4703 of *Lecture Notes in Computer Science*, pages 333–348. Springer, September 2007.

[EL09] Thomas Ehrhard and Olivier Laurent. Acyclic solos and differential interaction nets. Submitted to *Logical Methods in Computer Science*, 2009.

[ER06] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006.

[Ghi05] Dan Ghica. Slot games: A quantitative model of computation. In *Proceedings of the 32nd ACM Symposium on Principles of Programming Languages*, pages 85–97, 2005.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir91] Jean-Yves Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[Gir98] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, June 1998.

[Gir99] Jean-Yves Girard. On the meaning of logical rules I: syntax vs. semantics. In Ulrich Berger and Helmut Schwichtenberg, editors, *Computational Logic*, pages 215–272. Springer, 1999. NATO series F 165.

[Gir01] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, June 2001.

[Gri90] Timothy Griffin. A formulae-as-types notion of control. In *Proceedings of the 1990 Principles of Programming Languages Conference*, pages 47–58. IEEE, IEEE Computer Society Press, 1990.

[HL06] Russ Harmer and Olivier Laurent. The anatomy of innocence revisited. In S. Arun-Kumar and Naveen Garg, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 224–235. Springer, December 2006.

[HL08] Kohei Honda and Olivier Laurent. An exact correspondence between a typed pi-calculus and polarised proof-nets. Submitted to *Theoretical Computer Science*, November 2008.

[HO00] Martin Hyland and Luke Ong. On full abstraction for PCF. *Information and Computation*, 163(2):285–408, December 2000.

[Hug00] Dominic Hughes. *Hypergame Semantics: Full Completeness for System F*. Ph.D. thesis, Oxford University, 2000.

[HYB04] Kohei Honda, Nobuko Yoshida, and Martin Berger. Control in the π-calculus. In *Fourth ACM-SIGPLAN Continuation Workshop (CW'04)*, 2004. Online proceedings.

[Laf04] Yves Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 318(1–2):163–180, June 2004.

[Lai98] James Laird. *A semantic analysis of control*. Ph.D. thesis, University of Edinburgh, 1998.

[Lai01] James Laird. A fully abstract game semantics of local exceptions. In *Proceedings of the sixteenth annual symposium on Logic In Computer Science*, pages 105–114, Boston, June 2001. IEEE, IEEE Computer Society Press.

[Lau03] Olivier Laurent. Polarized proof-nets and $\lambda\mu$-calculus. *Theoretical Computer Science*, 290(1):161–188, January 2003.

[Lau05a] Olivier Laurent. Classical isomorphisms of types. *Mathematical Structures in Computer Science*, 15(5):969–1004, October 2005.

[Lau05b] Olivier Laurent. Syntax vs. semantics: a polarized approach. *Theoretical Computer Science*, 343(1–2):177–206, October 2005.

[Lau08] Olivier Laurent. Game semantics for first-order logic. Submitted to *Logical Methods in Computer Science*, November 2008.

[Lau09] Olivier Laurent. On the categorical semantics of elementary linear logic. *Theory and Applications of Categories*, 22(10):269–301, June 2009.

[LPV01] Cosimo Laneve, Joachim Parrow, and Björn Victor. Solo diagrams. In *Proceedings of the 4th conference on Theoretical Aspects of Computer Science, TACS'01*, volume 2215 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2001.

[LQTdF05] Olivier Laurent, Myriam Quatrini, and Lorenzo Tortora de Falco. Polarized and focalized linear and classical proofs. *Annals of Pure and Applied Logic*, 134(2–3):217–264, July 2005.

[LTdF06] Olivier Laurent and Lorenzo Tortora de Falco. Obsessional cliques: a semantic characterization of bounded time complexity. In *Proceedings of the twenty-first annual symposium on Logic In Computer Science*, pages 179–188, Seattle, August 2006. IEEE, IEEE Computer Society Press.

[LV03] Cosimo Laneve and Björn Victor. Solos in concert. *Mathematical Structures in Computer Science*, 13(5):657–683, 2003.

[Mel03] Paul-André Melliès. Categorical models of linear logic revisited. Prépublication électronique PPS//03/09//n°22 (pp), Laboratoire Preuves, Programmes et Systèmes, March 2003. To appear in *Theoretical Computer Science*.

[MO00] Andrzej Murawski and Luke Ong. Discreet games, light affine logic and PTIME computation. In *Computer Science Logic*, volume 1862 of *Lecture Notes in Computer Science*, pages 55–92. Springer, 2000.

[MT03] Harry Mairson and Kazushige Terui. On the computational complexity of cut-elimination in linear logic. In *Proceedings of the eighth Italian Conference on Theoretical Computer Science (ICTCS)*, volume 2841 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2003.

[Mur01] Andrzej Murawski. *On Semantic and Type-Theoretic Aspects of Polynomial-Time Computability*. Ph.D. thesis, Oxford University, 2001.

[Nic94] Hanno Nickau. Hereditarily sequential functionals. In Anil Nerode and Yuri Matiyasevich, editors, *Logical Foundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 1994.

[Pag09] Michele Pagani. The cut-elimination theorem for differential nets with boxes. In Pierre-Louis Curien, editor, *Typed Lambda Calculi and Applications '09*, volume 5608 of *Lecture Notes in Computer Science*. Springer, July 2009.

[Par92] Michel Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of International Conference on Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.

[Plo77] Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):225–255, 1977.

[PV98] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of the thirteenth annual symposium on Logic In Computer Science*, pages 176–185, Indianapolis, June 1998. IEEE, IEEE Computer Society Press.

[Red07] Brian Redmond. Multiplexor categories and models of soft linear logic. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science (LFCS)*, volume 4514 of *Lecture Notes in Computer Science*, pages 472–485. Springer, June 2007.

[Roc05] Jérôme Rocheteau. $\lambda\mu$-calculus and duality: Call-by-name and call-by-value. In Jürgen Giesl, editor, *Rewriting Techniques and Applications*, volume 3467 of *Lecture Notes in Computer Science*, pages 204–218. Springer, April 2005.

[See89] Robert Seely. Linear logic, $\star$-autonomous categories and cofree coalgebras. *Contemporary mathematics*, 92, 1989.

[Sel01] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, April 2001.

[TdF03] Lorenzo Tortora de Falco. Obsessional experiments for linear logic proof-nets. *Mathematical Structures in Computer Science*, 13(6):799–855, December 2003.

Résumé

La logique linéaire fait désormais partie des outils standards en théorie de la démonstration et, de manière plus générale, dans l'étude de la correspondance de Curry-Howard.

Nous présentons ici trois directions importantes d'application de méthodes issues de la logique linéaire :

- la théorie de la démonstration de la logique classique et ses aspects calculatoires via notamment la sémantique des jeux ;
- la complexité implicite à travers les modèles dénotationnels des logiques linéaires à complexité bornée ;
- la théorie de la concurrence et ses fondements logiques grâce aux ingrédients apportés par la logique linéaire différentielle.

Les approches linéaires offrent ainsi un cadre commun pour l'étude de différents aspects logiques du calcul.

Abstract

Linear Logic is now part of the toolbox for the development of proof theory as well as for the study of the Curry-Howard correspondence.

We present here three important directions for the application of methods coming from linear logic:

- proof theory for classical logic and its computational aspects using game semantics mainly;
- implicit computational complexity by means of the denotational models of light linear logic systems;
- concurrency theory and its logical foundations through the new ingredients provided by differential linear logic.

Linear Logic provides us this way with a common framework for the study of various logical aspects of computation.