Notice titres et travaux

Thomas Ehrhard
CNRS, PPS, UMR 7126
Univ Paris Diderot, Sorbonne Paris Cité
F-75205 Paris, France

e-mail : thomas.ehrhard@pps.univ-paris-diderot.fr
web : http://www.irif.univ-paris-diderot.fr/~ehrhard/

Septembre 2015

1 Informations personnelles

Nom:	Ehrhard
Prénom:	Thomas
Date et lieu de naissance:	27/04/1961, Strasbourg
Adresse personnelle:	150 avenue d'Italie, 75013 Paris
Téléphone personnel:	09 54 48 17 58
Téléphone mobile:	06 61 40 59 87
Téléphone professionnel:	01 57 27 92 17
Fax:	01 57 27 92 74
E-mail:	thomas.ehrhard@pps.univ-paris-diderot.fr
Web:	http://www.irif.univ-paris-diderot.fr/~ehrhard/
Etablissement:	CNRS
Grade:	Directeur de recherche au CNRS (DR1), section 06 (Informatique)

2 Titres universitaires

2.1 Doctorat

Université:	Paris 7
Spécialité:	Informatique
Directeur:	Pierre-Louis Curien
Titre:	Une sémantique catégorique des types dépendants. Application au calcul des constructions
Date et lieu de soutenance:	26 septembre 1988, Paris 7

2.2 Habilitation à diriger les recherches

Université:	Paris 7
Titre:	Stabilité forte et séquentialité
Date et lieu de soutenance:	26 janvier 1993, Paris 7

3 Curriculum vitae

1979-81:	Classes préparatoires au Lycée Kléber à Strasbourg.
1981:	Entrée à l'École Polytechnique.
1984:	Obtention du diplôme de l'École Polytechnique.
1984-85:	Exercice de la fonction d'ingénieur de développement dans une entreprise d'informatique à Strasbourg.
1985:	Entrée en fonction comme ingénieur système au Centre de Mathématiques de l'École Polytechnique, fonction que j'assure jusqu'en septembre 1989.
1985:	Inscription au DEA d'informatique fondamentale de Paris 7.
1986:	Obtention du diplôme de DEA et inscription en thèse d'informatique à Paris 7, sous la direction de Pierre-Louis Curien.
1988:	Le 26 septembre, soutenance de ma thèse.
1989-90:	En octobre 1989, entrée en fonction à l'université Paris 7 comme maître de conférences en informatique sur un poste fléché Marne-la-Vallée, université où j'ai exercé mes fonctions jusqu'en 94.
1993:	Le 26 janvier, soutenance de mon habilitation à diriger les recherches.
1994:	Entrée au CNRS, comme chargé de recherche de première classe (section 01), à l'Institut de Mathématiques de Luminy, à Marseille (IML, qui à l'époque s'appelait Laboratoire de Mathématiques Discrètes, LMD, et était une UPR du CNRS).
2001:	Passage au grade de directeur de recherche de seconde classe au CNRS (section 01).
2003:	À partir de septembre, responsable d'un « gros » projet ACI Nouvelles Interfaces des Mathématiques, intitulé GEOCAL. (Fin du projet en septembre 2006.)
2004:	À partir de janvier, responsable de l'équipe Logique de la Programmation de l'IML, jusqu'à septembre 2006.
2004:	À partir de septembre, responsable de la spécialité *Mathématiques discrètes et fondements de l'informatique* du Master 2 Mathématiques (université de la Méditerranée), jusqu'à septembre 2006.
2005:	À partir de septembre, membre à 75% du laboratoire *Preuves, Programmes et Systèmes* (UPR 7126, dirigé par Pierre-Louis Curien), et à 25% de l'IML, pour une durée de deux ans. Déménagement à Paris.
2007:	À partir de septembre, membre à 100% de PPS. Responsable du projet CHOCO (Curry-Howard pour la concurrence), projet «blanc» de l'ANR, durée 3 ans, accepté en juillet 2007. Le projet a commencé en novembre 2007 et a pris fin en avril 2011.
2009:	Depuis le 1er janvier, directeur du laboratoire *Preuves, Programmes et Systèmes*, UMR 7126, CNRS et université Paris Diderot — Paris 7.
2010:	Passage en section 07 du Comité National.
2011:	Passage à la 1ère classe des directeurs de recherche du CNRS.

Table des matières

4 Recherche

Cette partie décrit mes travaux de recherche depuis ma thèse de doctorat. Elle est assez longue (pp. 4–**??**). Je me suis efforcé de la diviser en sections pour en faciliter la lecture pas morceaux. Elle suit un ordre grossièrement chronologique (avec des exceptions lorsque l'unité thématique les rend nécessaires).

Pour les sujet sur lesquels je travaille activement en ce moment, j'ai ajouté la date approximative du début de l'activité correspondante. Voir par exemple la section 4.5 dont le titre comporte la mention «(depuis 2003)», puisque la logique linéaire et le lambda-calcul différentiel m'occupent depuis 2003, pratiquement sans interruption.

4.1 Introduction

Mon travail de recherche porte sur l'interprétation mathématique des processus calculatoires.

La notion de calcul a été dégagée au début du $20^{\text{ème}}$ siècle pour donner un sens au concept de faisabilité. Il s'agissait de discriminer ce qui est réalisable par un processus fini de ce qui ne l'est pas et de donner un sens mathématique précis à ce concept. Différentes caractérisations ont été obtenues, qui ont été montrées équivalente :

— lambda-calcul,
— schémas de Herbrand-Gödel,
— machines de Turing

pour ne citer que les plus anciens, mais il y en a bien d'autres qui se sont ensuite ajoutés à la liste.

Cela a conduit à la thèse de Church qui stipule l'équivalence entre tous les formalismes de description des fonctions partielles calculables

Avec l'apparition des ordinateurs, puis des langages de programmation, il est devenu évident que la simple distinction calculable/non calculable devenait insuffisante et que des outils mathématiques permettant de donner aux programmes une signification indépendante du formalisme dans lequel ils sont écrits étaient absolument nécessaires pour en dominer la complexité ; c'est ainsi qu'est née la *sémantique dénotationnelle*.

Les modèles dénotationnels partent souvent d'une propriété intuitivement évidente des programmes et lui donnent un sens mathématique précis dans un cadre catégorique où ils sont vus comme des morphismes. On cherche généralement à construire des catégories cartésiennes fermées

— car elles permettent d'interpréter les langages fonctionnels (Haskell, Ocaml etc),
— et les langages fonctionnels permettent d'interpréter les langages avec effets (grâce aux monades à la Moggi, par exemple)
— et surtout parce que c'est plus difficile, et donc plus exigeant d'un point de vue purement mathématique.

Ce dernier point est essentiel : la clôture cartésienne pose de très fortes contraintes sur les notions mathématiques que l'on introduit et nécessite de les raffiner et de les analyser en profondeur. La logique linéaire (voir plus loin), par les nouvelles symétries qu'elle introduit, renforce encore cette contrainte heuristique.

4.1.1 Continuité de Scott

Parmi les propriétés concrètes auxquelles s'intéresse la sémantique dénotationnelle, les deux plus simples sont les suivantes :

— Plus un programme dispose d'information à son entrée, plus il fournira d'information à sa sortie (l'absence d'information correspond au programme qui «boucle») ; par exemple, si un programme fournit un entier sur une entrée qui boucle, il fournira le même entier sur une entrée qui ne boucle pas.
— Un programme ne peut utiliser qu'une information finie pour produire une information finie ; par exemple, si un programme qui prend une fonction des entiers dans les entiers en argument fournit une certain résultat fini (disons un entier), il n'aura pu utiliser qu'une partie *finie* de la fonction fournie en argument, c'est-à-dire qu'il n'aura pu appliquer cette fonction qu'à un nombre fini d'entiers.

Ces propriétés sont à la base de la sémantique de Scott, qui a pour cadre des catégories d'ordres partiels [1] complets et de fonctions croissantes et Scott-continues (c'est-à-dire préservant les sups des familles filtrantes).

La sémantique dénotationnelle fait donc intervenir des objets partiels qui, s'ils sont parfaitement raisonnables du point de vue informatique (programmes qui bouclent), n'ont pas d'existence *a priori* en logique où les théorèmes d'élimination des coupures (normalisation faible ou forte) disent précisément que rien ne boucle. Ces objets partiels peuvent être éliminés par la suite au moyen de techniques de réalisabilité (on définit par récurrence sur les formules ce qu'est un objet «total», étant entendu que l'interprétation de toute preuve est totale), mais ils jouent un rôle essentiel dans la théorie : ils ont le même caractère canonique que les fonctions récursives partielles en théorie de la récursivité.

1. Où, intuitivement, un élément est d'autant plus grand qu'il est plus défini.

4.1.2 Stabilité

La sémantique *stable* (Berry [Ber78] et Girard [Gir86]) raffine la sémantique de Scott en partant de l'observation que, si un programme produit un entier n sur une fonction f donnée en argument, on peut dire qu'il existe une partie (finie) f_0 de f (une fonction donnant les mêmes valeurs que f sur les mêmes arguments, sauf peut-être pour certains, où elle est indéfinie) que le programme a effectivement utilisée pour produire son résultat. Autrement dit, le programme rend déjà n lorsqu'il est appliqué à f_0, et toute fonction moins définie que f sur laquelle le programme produit un résultat doit être plus définie que f_0.

Cela correspond à une sorte de déterminisme *a posteriori* : une fois le calcul terminé, on peut dire que la fonction passée en argument a été appliquée à un ensemble bien déterminé d'arguments.

Il existe des fonctions Scott-continues n'ayant pas cette propriété de stabilité, par exemple la fonction qui rend 0 dès que son argument (fonction des entiers dans les entiers) est défini sur 0 ou sur 1, et qui est indéfinie sinon. On voit bien qu'une telle fonction doit être capable de «deviner» si elle doit d'abord appliquer son argument à 0 ou à 1. On peut simuler ce comportement, au prix par exemple de l'introduction d'un opérateur parallèle, permettant d'évaluer simultanément $f(0)$ et $f(1)$. En tout cas, c'est impossible dans les langages «séquentiels [2]» comme ML, et, *a fortiori*, un tel comportement ne peut être réalisé par une preuve.

Propriété opérationnelle très naturelle, la stabilité a en plus le mérite de simplifier radicalement la sémantique dénotationnelle, comme l'a observé Girard. La notion générale d'ordre partiel complet mise en œuvre par la sémantique de Scott peut être remplacée, dans le cas stable, par la notion de graphe non étiqueté réflexif et symétrique (*espace cohérent* ou *espace de cohérence*) [3]. A partir d'un espace cohérent, on construit l'ordre partiel complet associé : c'est l'ensemble des cliques [4] (parties de la trame dont les éléments sont deux à deux cohérents) ordonné par l'inclusion. Une fonction stable d'un espace cohérent E dans un espace cohérent F est alors une fonction des cliques de E dans les cliques de F, croissante et continue, et commutant aux intersections des familles finies de cliques compatibles (au sens où leur réunion est encore une clique). De plus, toutes les constructions d'espaces requises par la sémantique dénotationnelle (produit cartésien, espaces de fonctions...) s'expriment très simplement en terme d'opérations sur les trames et les relations de cohérence. En particulier, la construction de l'espace cohérent $E \Rightarrow F$ des fonctions stables de E dans F peut se décomposer en deux opérations :

$$E \Rightarrow F = (!E) \multimap F$$

où $G \multimap F$ est l'espace des fonctions *linéaires* de G dans F (intuitivement, les fonctions qui utilisent leur argument exactement une fois), et $!E$ est une sorte d'«algèbre tensorielle symétrique» sur E (ce connecteur «!» et son dual [5] «?» sont appelés «exponentielles»). La découverte des espaces cohérents, et de cette décomposition de l'implication intuitionniste, a été déterminante dans la mise à jour de la logique linéaire par Girard.

2. L'expression «langage séquentiel» est à prendre au sens intuitif : il s'agit des langages où une seule opération a lieu à la fois.

3. Cette relation binaire est appelée cohérence, et l'ensemble des sommets du graphe est appelé la trame de l'espace cohérent.

4. Les éléments de la trame sont des informations atomiques, et la relation de cohérence dit si on peut les mettre ensemble pour former une donnée composée. Par exemple, on n'a pas le droit de mettre ensemble «vrai» et «faux» pour former une information booléenne.

5. Cette analyse fait aussi apparaître une notion de «négation» involutive, jouant exactement le même rôle que le passage au dual en algèbre linéaire : le passage au graphe complémentaire.

4.1.3 Sémantique relationnelle quantitative

Avant ce travail sur la stabilité, Girard s'était inspiré de ses travaux antérieurs sur les dilatateurs pour proposer une sémantique du lambda-calcul. Contrairement aux modèles évoqués jusqu'ici, ce travail de Girard introduisait pour la première fois l'idée d'une sémantique quantitative.

Il interprète les types par des ensembles, et un morphisme d'un ensemble I vers un ensemble J est un foncteur normal de $\mathbf{Set}^I$ vers $\mathbf{Set}^J$ (où $\mathbf{Set}$ est la catégorie des ensembles et des fonctions), c'est-à-dire un foncteur qui préserve les produites fibrés et les colimites filtrantes : ce sont les deux conditions qui définissent les dilatateurs, dans un autre contexte.

Il montre qu'un tel foncteur peut être décrit par un objet de la catégorie $\mathbf{Set}^{\mathcal{M}_{\mathrm{fin}}(I)\times J}$, qu'il faut voir comme une matrice dont les coefficients sont des ensembles (généralisant les nombres entiers), dont les colonnes sont indexées par $\mathcal{M}_{\mathrm{fin}}(I)$ (les multi-ensembles finis d'éléments de I) et les lignes par J. Cette décomposition préfigure celle que Girard a découverte ensuite dans les espaces cohérents et qui a donné naissance à la logique linéaire.

Je retrouverai ces idées plus tard, dans des modèles vectoriels où ces ensembles, qui jouent le rôle de coefficients potentiellement infinis, seront remplacés par des nombres.

4.1.4 De la syntaxe à la sémantique, et inversement

Cet aller-retour syntaxe-sémantique-syntaxe (ici : lambda-calcul - espaces cohérents - logique linéaire) est très fécond. Souvent, les spécialistes de ce domaine de recherche considèrent les théorèmes de complétude dénotationnel comme le Graal ultime : il s'agit de trouver des modèles dans lesquels tous les éléments sont définissables au moyen des termes de la syntaxe que l'on interprète. Une telle situation est appréciable, car elle donne des modèles qui permettent de raisonner de façon «complète» sur le langage considéré.

Mais, face à un modèle mathématiquement naturel, il est souvent plus fructueux d'essayer de comprendre «l'histoire que nous raconte ce modèle[6]», c'est-à-dire de chercher des syntaxes nouvelles rendant compte des structures nouvelles du modèle. C'est ce que Girard a fait avec la logique linéaire, à partir du modèle des espaces cohérents. Il a découvert de nouvelles constructions de types : la négation linéaire, le produit tensoriel et son opération duale (le par), les deux opérations exponentielles, duales l'une de l'autre.

Pour ma part, je me suis plusieurs fois livré à cette activité et c'est ainsi que j'ai introduit le lambda-calcul différentiel. C'est l'une des principales raisons pour lesquelles je m'intéresse à la sémantique dénotationnelle : c'est une source d'inspiration irremplaçable.

4.2 Séquentialité

Je commence maintenant à décrire mes propres contributions. Mes travaux de doctorat ont porté sur la sémantique catégorique du calcul des constructions (Coquand-Huet), qui sont indépendants de ce que j'ai fait par la suite. Je ne les évoquerai donc pas ici pour ne pas alourdir ce texte déjà trop long.

La stabilité (voir 4.1.2) ne parvient pas à capturer le caractère séquentiel des langages de programmation et des preuves. Ainsi, une fonction à trois arguments entiers et à résultat entier rendant

6. Pour reprendre approximativement les termes d'une question de Pierre-Louis Curien à la soutenance de mon HDR.

0 dès que son argument (un triplet) est plus défini que $(\bot, 0, 1)$, $(1, \bot, 0)$ ou $(0, 1, \bot)$ [7] et indéfinie autrement, est parfaitement stable, mais ne peut pas être programmée dans un langage séquentiel : lequel de ses trois arguments devrait-elle tester en premier (il faut bien qu'il y en ait un) ?

En cherchant à caractériser la séquentialité des langages fonctionnels en sémantique dénotationnelle, Berry et Curien ont été amenés à la fin des années 70 [8] à introduire les *algorithmes séquentiels*, que l'on comprend mieux aujourd'hui comme des stratégies déterministes dans certains jeux à deux joueurs. Sans entrer dans les détails, on peut donner une intuition sur l'interprétation comme algorithme séquentiel d'un programme F de type $(\iota \to \iota) \to \iota$, qui prend en argument une fonction des entiers dans les entiers, et rend un entier. Voici les comportements possibles d'un tel algorithme séquentiel :

- Il peut être totalement indéfini. Autrement dit, il boucle et ne fait absolument rien, ni à son entrée, ni à sa sortie.
- Il peut rendre directement un entier, sans consulter son entrée (algorithme constant).
- Il peut avoir besoin de son argument (un algorithme séquentiel f des entiers dans les entiers). Il demande alors à la sortie de son argument une valeur. L'argument f a aussi trois comportements possibles :
  - Il peut boucler sans produire de résultat ni consulter son entrée, et dans ce cas l'application de F à f est indéfinie.
  - Il peut fournir directement un entier (f est un algorithme constant), et alors F est satisfait et reprend ses calculs qui peuvent le mener soit à fournir un résultat à la sortie, soit à redemander à f sa valeur [9], ou encore à boucler.
  - Ou alors, pour produire un entier à sa sortie, f a besoin de connaître son argument, qu'il demande à F. Alors, F peut soit boucler, soit fournir à f un entier en argument [10]. Si f boucle et ne rend rien, l'ensemble du processus d'application de F à f boucle. Sinon, f produit un entier, qu'il donne à F en réponse à la question qu'il lui avait posée. Alors F reprend ses calculs, et peut être amené soit à fournir un résultat à la sortie, soit à boucler, soit à interroger à nouveau son argument f, qui aura besoin d'un argument, que F lui fournira (une valeur en général différente de celle fournie à l'étape précédente), et le dialogue continue ainsi...

4.2.1 Algorithmes séquentiels abstraits

En 1989, avec Antonio Bucciarelli (Paris 7, laboratoire PPS), nous avons commencé à travailler sur les algorithmes séquentiels, et notre première contribution a été de fournir une description plus abstraite de la catégorie des algorithmes séquentiels ([BE93]). Ce travail s'est révélé précieux par la suite.

7. Où $\bot$ désigne la donnée entière indéfinie, c'est-à-dire, intuitivement, le programme de type entier qui boucle indéfiniment.

8. Des travaux similaires étaient menés de façon contemporaine par Kleene dans la tradition de la théorie des fonctions récursives, mais ils n'aboutirent pas à des résultats aussi satisfaisants.

9. Ce comportement est redondant, puisqu'on connaît déjà la valeur que rendra f. Il n'apparaît pas dans les algorithmes séquentiels, mais dans d'autres sémantiques de jeux plus fidèles à la syntaxe.

10. Dans les algorithmes séquentiels, à ce stade, F a également le droit de fournir un entier à la sortie, achevant ainsi le dialogue. Ce comportement ne correspond à rien dans les langages purement fonctionnels, ou dans les preuves, mais peut parfaitement être simulé au moyen d'exceptions, dans ML par exemple : c'est un comportement séquentiel, mais non fonctionnel.

4.2.2 Stabilité forte et domaines qualitatifs

En cherchant encore à simplifier cette théorie, nous nous sommes rendu compte que, pour les fonctions de types de base dans des types de base (disons, pour fixer les idées, de ι^k dans ι^p, où ι est le type des entiers), la notion de séquentialité, qui est connue depuis le début des années 70 grâce aux travaux indépendants de Vuillemin, Milner et Sazonov (VMS dans la suite), pouvait être traduite en une condition de préservation d'une certaine notion de cohérence (plus générale que celle de la sémantique stable), et de commutation aux infs des familles cohérentes en ce sens : la condition de *stabilité forte*.

Alors que la séquentialité de VMS ne s'étend absolument pas aux espaces de fonctions séquentielles [11], cette condition beaucoup plus souple de stabilité forte s'étend sans grande difficulté aux ordres supérieurs. Cela nous a amenés à introduire une nouvelle sémantique des langages fonctionnels et de la logique intuitionniste : la sémantique fortement stable ([BE91b]).

Nous pensions alors pouvoir par ce moyen produire un modèle dénotationnellement complet des langages purement fonctionnels, mais nous savions que cela exigeait un raffinement de la sémantique fortement stable au moyen de la sémantique de Scott, que nous avons mis au point dans [BE91a] au moyen de *relations logiques* (la *réductibilité* des logiciens). Le modèle ainsi obtenu, même s'il n'est malheureusement pas dénotationnellement complet, raffine des tentatives antérieures de Berry dans cette direction (bidomaines, dans le cas stable bien sûr), et a une théorie équationnelle plus grande (au sens de l'inclusion) que celle du modèle de Scott, comme nous l'avons montré dans [BE94].

4.2.3 Hypercohérences et collapse extensionnel

Depuis la mise au point de la sémantique fortement stable avec Bucciarelli, la question de savoir quelle pouvait être la signification opérationnelle de la stabilité forte aux types plus fonctionnels que $\iota^k \to \iota$ était ma principale préoccupation (dans la perspective méthodologique expliquée en 4.1.4). Je m'étais peu à peu convaincu qu'une connexion étroite devait exister *à tous les types* entre la hiérarchie des algorithmes séquentiels et celle des fonctions fortement stables. C'est en cherchant à expliciter cette connexion que j'ai trouvé un cadre très simple, et très similaire aux espaces cohérents, pour la sémantique fortement stable : les *hypercohérences* ([Ehr95]). Une hypercohérence est un hypergraphe réflexif et symétrique, c'est-à-dire un ensemble (la trame) muni d'un sous-ensemble de l'ensemble de ses parties finies non vides contenant tous les singletons (les parties cohérentes). On voit bien la différence par rapport aux espaces cohérents : il s'agit toujours d'atomes munis d'une notion de cohérence, mais ici, cette notion n'est plus binaire (on ne peut d'ailleurs pas poser de borne à son arité). Comme dans le cas des espaces cohérents, on peut associer à une hypercohérence un ensemble de cliques, naturellement ordonné par l'inclusion : une clique est une partie de la trame dont tous les sous-ensembles finis et non vides sont cohérents. Les cliques finies sont donc les simplexes d'un «complexe simplicial» dont les sommets sont les éléments de la trame [12]. On peut ensuite définir une notion de cohérence sur les cliques d'une hypercohérence, et utiliser cette notion de cohérence pour définir les fonctions fortement stables d'une hypercohérence dans une autre : ce

11. C'est ce phénomène qui a poussé Berry et Curien à introduire les algorithmes séquentiels, remplaçant les fonctions séquentielles par des stratégies décrivant explicitement l'ordre dans lequel elles explorent leur argument.

12. Il y a des «trous», par exemple, dans l'interprétation hypercohérente du type $\iota^3 \to \iota$, les trois points de la trame que sont les trois fonctions «atomiques» f_1, f_2 et f_3 données par $f_1(x,y,z) = 0$ si (x,y,z) est plus défini que $(\bot, 0, 1)$ et $f_1(x,y,z) = \bot$ sinon, et de même (permutation circulaire sur les arguments) pour f_2 et f_3, sont deux à deux cohérentes (en effet, par exemple la fonction $f_{1,2}(x,y,z) = 0$ si $f_1(x,y,z) = 0$ ou $f_2(x,y,z) = 0$ est séquentielle), mais pas globalement cohérentes. Les espaces cohérents ne peuvent évidemment pas rendre compte de ce genre d'effet.

sont les fonctions continues de l'espace des cliques de la première hypercohérence dans celui de la seconde, qui préservent la cohérence et commutent aux intersections des familles cohérentes. Comme dans le cas des espaces cohérents, toutes les constructions nécessaires à la sémantique s'expriment très simplement en termes d'opérations sur les trames et les relations de cohérence.

De plus, et c'était une surprise de taille, j'ai constaté que les hypercohérences forment un modèle de la logique linéaire. Cela m'a définitivement convaincu de l'importance fondamentale de cette logique, puisqu'elle apparaît ainsi spontanément, sans être convoquée. J'ai constaté ce phénomène à de nombreuses reprises par la suite.

Collapses extensionnels des algorithmes séquentiels. Ce nouveau cadre m'a permis de démontrer des théorèmes reliant la sémantique fortement stable aux jeux (algorithmes séquentiels).

Dans [Ehr96], j'ai utilisé le cadre abstrait introduit dans [BE93] pour montrer que, dans la hiérarchie des types finis (ceux qu'on peut construire en utilisant le type de base ι des entiers naturels et l'opérateur «$\to$»), à condition de poser certaines restrictions sur les algorithmes séquentiels considérés, on pouvait, pour chaque type, établir une projection (surjective) de l'espace des algorithmes séquentiels de ce type sur l'espace des fonctions fortement stables de ce même type. Cette projection consiste à oublier *héréditairement* toutes les informations explicites de temporalité contenues dans les stratégies pour ne retenir que leur comportement extensionnel. Ce résultat était une bonne indication que, du moins dans cette hiérarchie de types, toute fonction fortement stable est «séquentiellement réalisable».

J'ai obtenu un résultat de même nature en montrant, toujours dans la hiérarchie des types finis, que toute fonction fortement stable (finie) peut être obtenue en appliquant un programme *purement fonctionnel* à une fonction fortement stable dont le type est de la forme $(\iota^k \to \iota) \to \iota$ (pour un entier k dépendant de la fonction dont on est parti). Il est facile de se rendre compte que, pour les types de cette forme, toute fonction fortement stable est réalisable par un algorithme séquentiel, et ce théorème de définissabilité relative permet d'étendre cette propriété à tous les types. Enfin, j'ai déduit de tout cela que, dans la hiérarchie des types finis, le modèle des fonctions fortement stables est isomorphe au *collapse extensionnel*[13] du modèle des algorithmes séquentiels.

En plus de fournir un argument en faveur de la thèse selon laquelle les fonctions fortement stables sont séquentielles à tous les types, ce théorème fournit une caractérisation indépendante de la hiérarchie des fonctions fortement stables (la définition des algorithmes séquentiels est très différente de celle des fonctions fortement stables). Or c'est toujours bon signe, me semble-t-il, quand une même notion mathématique peut être caractérisée de plusieurs façons différentes.

Des caractérisations similaires de la hiérarchie fortement stable ont ensuite été obtenues par Van Oosten (université d'Utrecht) et Longley (université d'Edimbourg), voir par exemple le long papier

13. Un modèle est *extensionnel* si ses morphismes sont complètement caractérisés par leur comportement applicatif, c'est-à-dire par la valeur qu'ils rendent quand on les applique à des arguments, autrement dit, si la catégorie correspondante *a assez de points*. Ce n'est pas le cas des modèles de jeux (puisque les stratégies explicitent complètement l'ordre dans lequel les arguments sont explorés), alors que c'est le cas des modèles continus, stables et fortement stables. Le collapse extensionnel consiste à identifier, dans un modèle non extensionnel, les morphismes ayant le même comportement applicatif (tout à fait comme dans l'opération homonyme de théorie des ensembles). Plus précisément, on munit chaque type d'une relation d'équivalence partielle définie comme l'égalité au type de base ι, et au type $\sigma \to \tau$ en disant que f est équivalent à g si à chaque fois que x est équivalent à y (dans σ), $f(x)$ est équivalent à $g(y)$ (dans τ). La relation ainsi obtenue est partielle, car pour qu'un morphisme f soit auto-équivalent, il faut qu'il préserve la relation d'équivalence partielle, ce qui n'est pas forcément le cas. Le collapse du modèle s'obtient en quotientant les éléments auto-équivalents de l'interprétation de chaque type par la relation d'équivalence partielle qui lui est ainsi associée.

de ce dernier auteur consacré aux «fonctions séquentiellement réalisables» (c'est-à-dire fortement stables[14]), [Lon98].

4.2.4 Caractérisation des fonctions fortement stables par préfaisceaux.

Avec Colson (université Paris 7, à l'époque), nous avons obtenu une autre caractérisation de la hiérarchie fortement stable, à base de réductibilité. On sait ce que signifie être séquentiel pour une fonction de type $\iota^n \to \iota$, pour tout entier naturel n (définition de VMS). Voici une définition naturelle pour la séquentialité d'une fonctionnelle f de type $(\iota^k \to \iota) \to \iota$: pour tout entier naturel n, pour toute fonction séquentielle $g : \iota^{n+k} \to \iota$, la fonction $f \circ g' : \iota^n \to \iota$ est séquentielle, où $g' : \iota^n \to (\iota^k \to \iota)$ est la fonction qui au n-uple x associe la fonction qui au k-uple y associe $g(x, y)$. Cette idée peut être étendue, par récurrence, à tous les types. Nous avons prouvé que cette nouvelle notion de séquentialité est en fait équivalente à la stabilité forte ([CE94]), ce qui fournit une troisième caractérisation indépendante de cette notion.

Ce type de définition peut être appliqué dans bien d'autres cas que la séquentialité, et peut être reformulé comme une construction générale de *préfaisceaux extensionnels* sur une catégorie ayant suffisamment de points, construction que l'on reverra plus loin, dans le cadre des *espaces de finitude*, voir 4.4.2, p. 18.

4.2.5 Hypercohérences séries et parallèles

Il s'agit ici d'un travail qui visait à reconstruire les jeux à partir des hypercohérences en partant de l'observation suivante. Si on définit une hypercohérences comme *parallèle* quand la réunion de deux parties cohérentes qui se rencontrent est encore cohérente, et comme *série* si sa négation linéaire est parallèle, on se rend compte que les hypercohérences qui sont à la fois série et parallèle peuvent être décrites comme des espaces cohérents très simples, dits «sans P4», pour reprendre la terminologie de la théorie des graphes. Un tel espace cohérent peut être décrit comme l'ensemble des positions finales d'un jeu déterministe à deux joueurs, deux points étant cohérents si c'est l'opposant qui est responsable de la divergence entre les parties correspondantes, et dualement pour l'incohérence.

J'ai défini une opération générale qui permet d'associer à chaque hypercohérence une hypercohérence parallèle, avec une projection sur l'hypercohérence de départ, une sorte de revêtement caractérisé par une propriété de rigidité (pas d'automorphismes non triviaux). Cette construction semble être un outil naturel pour reconstruire, à partir d'une hypercohérence quelconque, le jeu déterministe dont elle est le collapse extensionnel selon mes travaux antérieurs 4.2.3. Ses propriétés de base sont explorées dans [Ehr00].

4.3 Logique linéaire indexée

Les relations logiques sont un outil fondamental pour raisonner sur les programmes et pour en prouver des propriétés opérationnelles. On en a déjà mentionné deux exemples : totalité et collapse extensionnel.

En sémantique dénotationnelle, cette technique permet de raffiner les modèles, et de se rapprocher de situations de complétude dénotationnelle.

14. Il n'y a pas ici de querelle de terminologie, le terme «fortement stable» étant effectivement, *a posteriori*, trop neutre. Mais à l'époque (1991), on ne pouvait pas être sûr que les fonctions fortement stables étaient séquentiellement réalisables...

Dans le cadre intuitionniste habituel (catégorie cartésiennes fermées etc), une relation logique I-aire (où I est un ensemble d'indices) est la donnée, pour chaque type A, d'une relation I-aire sur les objets de type A, avec la contrainte qu'une famille de «fonctions» $(f_i)_{i\in I}$ de type $A \Rightarrow B$ est dans la relation si et seulement si, pour toute famille $(a_i)_{i\in I}$ d'éléments de type A qui est dans la relation, la famille $(f_i(a_i))_{i\in I}$ d'éléments de type B est dans la relation.

La notion s'étend à la logique linéaire, avec la contrainte supplémentaire que, à chaque type, la relation logique doit satisfaire une condition de clôture correspondant à l'identification du type avec sa double négation linéaire.

4.3.1 Relations logiques et logique linéaire

Cependant, pour obtenir des résultats de complétude dénotationnelle, il était déjà apparu dans les travaux de Girard [Gir99] qu'il convient d'enrichir le cadre sémantique ordinaire, qui est purement qualitatif. Avec Bucciarelli, à l'occasion de son invitation d'un mois à l'IML en février 1998, nous nous sommes intéressés au pouvoir expressif de cette technique au sein du fragment additif-multiplicatif de la logique linéaire (MALL) [15]. En enrichissant le cadre sémantique habituel au moyen d'un monoïde commutatif, nous sous sommes rendu compte qu'il y a un rapport étroit entre relations logiques et sémantique des phases [16]. Plus précisément, si A est une formule de MALL, on peut associer à toute famille α de points de la trame [17] de A une formule $A(\alpha)$ dans une logique qui est une extension [18] de MALL, en sorte que la famille α soit contenue dans l'interprétation d'une preuve de A si et seulement si la formule $A(\alpha)$ est prouvable dans cette nouvelle logique. Cette nouvelle *logique indexée* a elle-même une sémantique des phases (que nous avons dérivée des relations logiques : c'est là un point de contact surprenant entre sémantique de vérité et sémantique dénotationnelle), et on peut prouver un théorème de complétude [19] pour cette sémantique, d'où l'on peut ensuite déduire un théorème de complétude dénotationnelle. Ces premiers résultats sont présentés dans [BE00].

4.3.2 Localisation des exponentielles

Nous avons ensuite étendu ces constructions à l'ensemble de la logique linéaire propositionnelle du premier ordre, exponentielles comprises. Nous avons ainsi défini une logique linéaire indexée, où les exponentielles sont décorées par des fonctions. Chaque formule a un domaine (qui est un ensemble) et si A est une formule de domaine J et u une fonction «presque injective» (i.e. : qui préserve les ensembles finis par image inverse) de J dans K, $!_u A$ et $?_u A$ sont des formules de domaine K. Si l'on admet que A représente une famille J-indexée α d'éléments de la trame de la formule de la logique linéaire S sous-jacente à A, alors il faut considérer $!_u A$ et $?_u A$ comme représentant la famille K-indexée de multi-ensembles finis d'éléments de la trame de S dont le k-ième élément est le multi-ensemble [20] des éléments de la famille α restreinte aux indices appartenant à $u^{-1}(\{k\})$

15. Déjà dans ce fragment, ni les espaces cohérents, ni les hypercohérences ne fournissent de sémantique dénotationnellement complète.

16. La sémantique des phases est la sémantique de vérité de la logique linéaire, de même que la sémantique booléenne est la sémantique de vérité de la logique classique.

17. C'est-à-dire de la trame de l'espace cohérent associé à A, ou de l'hypercohérence associée à A : dans MALL, la cohérence n'intervient pas dans la construction des trames, il n'y a couplage entre cohérence et trame qu'à partir du moment où l'on veut interpréter aussi les exponentielles.

18. Les formules de MALL correspondent dans cette extension aux familles vides.

19. Un théorème de complétude énonce que ce qui est vrai dans tous les modèles est prouvable.

20. Car on tient compte des multiplicités.

(ensemble fini car u est presque injective). Comme la trame de $!S$ et de $?S$ est justement l'ensemble des multi-ensembles finis d'éléments de la trame de S, on a bien associé à $!_u A$ et à $?_u A$ une famille K-indexée d'éléments de la trame de la formule de la logique linéaire sous-jacente à ces formules.

4.3.3 Logique linéaire indexée et sémantique des phases

Toutes les constructions de la logique linéaire indexée admettent une interprétation naturelle dans les *espaces de phase produit*. Un ensemble global, infini dénombrable, d'indices I étant fixé, un espace de phase produit est une structure de la forme $(P_0^I, \perp)$ où P_0 est un monoïde commutatif (noté multiplicativement) possédant un élément absorbant 0, P_0^I est le monoïde produit et $\perp$ est un sous-ensemble de P_0^I sujet à deux conditions de clôture, dont l'une exprime essentiellement son invariance par toute permutation de I (symétrie). On dit alors que deux éléments $p, q \in P_0^J$ (pour $J \subseteq I$) sont orthogonaux si le produit $pq = (p_i q_i)_{i \in J}$ admet une extension dans $\perp$. Une formule de domaine J est interprétée par un sous-ensemble de P_0^J qui est égal à son bi-orthogonal (l'orthogonal d'une partie U de P_0^J étant l'ensemble des éléments de P_0^J qui sont orthogonaux à tous les éléments de U) : en sémantique des phases, un tel ensemble est appelé un *fait*.

L'interprétation des exponentielles utilise une correspondance fonctorielle : à une fonction presque injective u de J dans K, on peut associer le morphisme de monoïde $u_* : P_0^J \to P_0^K$ qui à une famille $p \in P_0^J$ associe la famille K-indexée dont le k-ième élément est le produit fini $\prod_{j \in u^{-1}(\{k\})} p_j$. On obtient ainsi une sémantique de vérité pour la logique linéaire indexée, dont on a observé qu'elle n'est pas complète, pour des raisons qui semblent intéressantes (j'y reviendrai plus loin).

On a aussi montré comment associer à chaque espace de phase produit une sémantique dénotationnelle de la logique linéaire. Tous les modèles ainsi obtenus associent la même trame à chaque formule (à savoir son interprétation relationnelle, voir 4.7), mais cette trame est équipée d'une opération naturelle (au sens catégorique) associant à chaque famille J-indexée (pour tout $J \subseteq I$) un fait de P_0^J dans l'espace de phase produit considéré. En considérant des cas particuliers très simples, on a obtenu de cette façon une nouvelle version de la sémantique cohérente de la logique linéaire, qui diffère radicalement de la sémantique usuelle dans l'interprétation des exponentielles. Ces espaces cohérents sont non uniformes au sens où un point de la trame n'est pas forcément cohérent avec lui-même. Je retrouverai cette idée de cohérence non uniforme dans mon travail sur le lambda-calcul avec ressources, voir 4.5.2.

Tout ce travail fait l'objet du papier [BE01]. On trouvera dans des travaux antérieurs de François Lamarche (INRIA, Nancy) et de Glynn Winskel (université de Cambridge), qui ont généralisé les hypercohérences dans deux directions différentes, des idées qui anticipent certains aspects de nos constructions.

4.3.4 Incomplétude et... complétude

Comme on l'a déjà dit, les espaces de phase produit ne fournissent pas de sémantique complète de la logique linéaire indexée définie dans [BE01]. La raison essentielle de cette incomplétude est que, dans les espaces de phase produit, les formules A et $!_{\mathrm{Id}} A$ (où Id est la fonction identité du domaine de A dans lui-même) ont la même interprétation. Dénotationnellement, cela correspond au fait que, dans la catégorie associée à un espace de phase produit, tout objet X est un sous-espace de $!X$. Si bien que d'un morphisme $!X \multimap Y$, il est possible d'extraire une «composante linéaire» $X \multimap Y$ (une sorte de «dérivée en 0» : voir plus loin le développement de cette idée dans le cadre du lambda-calcul et de la logique linéaire différentiels, voir 4.5.). Dans [Ehr04], je définis une extension de la logique

linéaire indexée (avec les mêmes formules) dans laquelle la formule $A \multimap !_{\mathrm{Id}} A$ est toujours prouvable. Cette logique est présentée comme un calcul des séquents, et je prouve un théorème de complétude pour les espaces de phase produit vis-à-vis de ce calcul des séquents. Un des intérêts de ce théorème de complétude est de marquer la frontière du pouvoir expressif des modèles dénotationnels induits par les espaces de phases produit. D'autre part, il reste un espoir de démontrer un théorème de complétude pour les espaces de phase produit symétriques par rapport à la logique linéaire indexée ordinaire en renforçant la sémantique par une contrainte de *totalité*.

4.3.5 Sémantique relationnelle

Ces travaux m'ont amené à considérer la sémantique relationnelle de la logique linéaire comme absolument fondamentale. Cette sémantique très élémentaire fait partie du folklore de la logique linéaire depuis de nombreuses années. Elle consiste à associer aux formules des ensembles, et aux preuves des relations entre ces ensembles. Elle est sous-jacente à la sémantique quantitative (4.1.3) de la logique intuitionniste introduite par Girard dans [Gir88], et aussi à la sémantique des espaces cohérents, des hypercohérences (4.2.3) etc.

Je consacre la section 4.7 à mes travaux dans ce domaine, qui sont plus récents.

4.4 Modèles vectoriels de la logique linéaire

Dans la suite de ce travail, j'ai naturellement été amené à considérer des modèles de la logique linéaire «quantitatifs» au sens où les preuves sont interprétées par des vecteurs à valeur dans un corps. De ce point de vue, les éléments de la sémantique relationnelle d'une formule doivent être vus comme des vecteurs de base, pour une base particulière. Un type est alors interprété comme un espace vectoriel (muni d'une base particulière), et une preuve comme une application linéaire.

4.4.1 Espaces de Köthe

J'ai étudié le premier modèle de ce type en 2000 : les espaces de suites de Köthe. Ce sont des espaces vectoriels topologiques localement convexes qu'on peut décrire comme suit : on considère un ensemble (au plus dénombrable) I, et on dit que deux éléments u et u' de $\mathbb{R}^I$ sont en dualité si la famille $uu' = (u_i u'_i)_{i\in I}$ est absolument sommable. Un sous-ensemble E de $\mathbb{R}^I$ qui est égal à son bi-dual est alors automatiquement un $\mathbb{R}$-sous espace vectoriel de $\mathbb{R}^I$. De plus, à chaque élément u' de $E^\perp$ (les $u' \in \mathbb{R}^I$ qui sont en dualité avec tous les éléments de E), on peut associer une semi-norme N sur E en posant $N(u) = \sum_{i\in I} |u_i u'_i|$. Muni de la topologie induite par ces semi-normes, E est alors un evt localement convexe complet, dans lequel la famille «canonique» usuelle $(e_i)_{i\in I}$ (avec $(e_i)_j = \delta_{i,j}$, le symbole de Kronecker) constitue une base de Schauder. Les espaces qui interprètent les formules dont donc des paires $X = (|X|, E_X)$ où $|X|$ est un ensemble au plus dénombrable, et E_X un sous-espace de $\mathbb{R}^{|X|}$ qui est égal à son bi-dual, et qui de ce fait a une structure canonique d'evt localement convexe (evtlc) complet. J'appelle ces paires *espaces de Köthe*. Les preuves sont interprétées par des vecteurs dans les espaces correspondants.

Tous les connecteurs de la logique linéaire s'interprètent comme des opérations très naturelles sur ces espaces : la négation linéaire correspond au passage au dual topologique (qui est ici une opération involutive, contrairement par exemple à ce qui se passe avec les Banach [21]), l'implication

21. Par exemple, le Banach l^1 est un espace de Köthe, et son dual est l'espace des familles bornées, mais muni d'une topologie strictement plus faible que celle du Banach l^∞, et qui ne peut pas être décrite au moyen d'une unique

linéaire correspond à la construction de l'espace des fonctions linéaires continues, le «fois» (produit tensoriel) de la logique linéaire correspond à un produit tensoriel, les opérations additives correspondent aux opérations de produit et somme directs. Enfin, l'exponentielle $!X$ d'un tel espace est le dual topologique d'un espace de fonctions «entières» de E_X dans $\mathbb{R}$, muni d'une certaine topologie. Noter que ces fonctions, bien qu'entières (et donc analytiques), ne sont pas continues relativement à la topologie canonique de E_X en général (cf. par exemple le rôle de la notion d'hypocontinuité pour les fonctions multilinéaires dans la théorie générale des evtlc). Plus généralement, l'implication intuitionniste (qui se décompose en exponentielle et implication linéaire) correspond à la construction d'un espace de fonctions entières d'un espace de Köthe dans un autre : on a ainsi construit une catégorie cartésienne fermée dont les morphismes sont des fonctions entières.

Ces constructions restent très simples grâce à la présence de la base canonique $|X|$ dans la donnée d'un espace de Köthe X. Cela serait conceptuellement gênant si les diverses constructions d'espaces dépendaient de ces bases. On montre, également dans [Ehr02], que ce n'est pas le cas en introduisant une notion d'espace de Köthe *abstrait* qui, par définition, est un evt linéairement homéomorphe à E_X pour au moins un espace de Köthe X (cela signifie simplement que E est un evtlc complet possédant une base absolue ayant en plus une propriété particulière), et en montrant que toutes les constructions d'espace évoquées ci-dessus peuvent être réalisées sur les espaces de Köthe abstraits. Cette théorie «intrinsèque» reste toutefois assez lacunaire : il manque essentiellement des théorèmes caractérisant, *sans faire intervenir les bases*, les topologies associées à ces constructions d'espace. Il n'est pas clair du tout non plus que cette classe d'espaces localement convexe soit stable sous certaines opérations importantes, comme le passage au sous-espace fermé, ou les quotients.

4.4.2 Espaces de finitude

J'ai mis au point une version «discrète» des espaces de Köthe : les *espaces de finitude*, qui font l'objet de [Ehr05]. La définition est proche de celle des espace de Köthe, la principale différence est qu'on remplace «famille absolument sommable» par «famille finie», ce qui a l'avantage de rendre la notion indépendante du corps des scalaires. On peut donc définir d'abord les espaces de finitude comme des objets purement combinatoires (formellement proches des espaces cohérents) qui, dans un deuxième temps, permettent de définir des espaces vectoriels topologiques sur n'importe quel corps. Dans la version combinatoire comme dans la version vectorielle, on obtient un modèle de la logique linéaire.

Version combinatoire. Un espace de finitude est un couple $X = (|X|, \mathsf{F}(X))$ où $|X|$ est un ensemble au plus dénombrable (la trame de X), et $\mathsf{F}(X)$ est un sous-ensemble de $\mathcal{P}(|X|)$, les *parties finitaires* de $|X|$, qui doit être égal à son bi-dual pour la notion de dualité «avoir une intersection finie». Autrement dit, $\mathsf{F}(X)$ doit être l'ensemble de toutes les parties de $|X|$ qui ont une intersection finie avec toutes les parties de $|X|$ qui ont une intersection finie avec tous les éléments de $\mathsf{F}(X)$.

Le dual $X^\perp$ de X est l'espace de finitude dont la trame $|X^\perp|$ est $|X|$ et où un sous-ensemble de cette trame est finitaire s'il a une intersection finie avec tout élément de $\mathsf{F}(X)$, donc $X^{\perp\perp} = X$. On définit un produit tensoriel $X \otimes Y$ avec $|X \otimes Y| = |X| \times |Y|$, un «espace des morphismes linéaires» $X \multimap Y = (X \otimes Y^\perp)^\perp$, un produit direct (qui coïncide avec la somme directe) $X \,\&\, Y$ dans lequel $|X \,\&\, Y|$ est l'union disjointe de $|X|$ et $|Y|$ et une exponentielle $!X$ dans laquelle $|!X|$ est l'ensemble

norme.

des multi-ensembles finis d'éléments de $|X|$, et on obtient ainsi un nouveau modèle dénotationnel de la logique linéaire.

J'ai observé que ce modèle n'interprète pas l'opérateur de point fixe Y de PCF (une version du schéma μ de la théorie des fonctions récursives), mais qu'on peut néanmoins y donner une sémantique à la récursion primitive (et donc au système T de Gödel), et même au système F de Girard (calcul propositionnel intuitionniste du second ordre), qui est beaucoup plus puissant en terme de fonctions définissables. Ce dernier résultat n'a pas encore fait l'objet d'une publication.

Cette sémantique semble donc refléter des propriétés de normalisation de la syntaxe, auquel cas l'ensemble vide (qui est présent dans la sémantique de tous les types) devrait être vu non pas comme le programme qui boucle (intuition standard dans les sémantiques «à la Scott», voir 4.1.1), mais plutôt comme une sorte de démon, au sens de la *ludique* de Girard, c'est-à-dire de terminaison sans résultat.

Version vectorielle. En 1942, Lefschetz [Lef42] introduit la notion d'*espace vectoriel à topologie linéaire* (on dira simplement *ltvs*). Ce sont des espaces vectoriels topologiques qui sortent du cadre localement convexe habituel, car ils considèrent le corps comme muni de la topologie discrète ; on peut donc avoir des ltvs sur n'importe quel corps, pas seulement celui des réels ou des complexes. La topologie d'un ltvs est engendrée par un filtre de voisinage de 0 constitué de sous-espaces vectoriels, ce qui permet de retrouver une propriété de Hahn-Banach. Ils ont des propriétés assez différentes des espaces localement convexes, par exemple, ils sont totalement discontinus. Mais, *mutatis mutandis*, il semble correct de les voir comme un analogue purement algébrique des espaces localement convexes. Les espaces de finitude ont un lien direct avec les ltvs.

Soit $\mathbf{k}$ un corps, choisi une fois pour toutes. Étant donné un espace de finitude X, l'ensemble $\mathsf{F}(X)$ est clos par unions finies et il est donc possible de définir l'espace vectoriel $\mathbf{k}\langle X\rangle$ des éléments de $\mathbf{k}^{|X|}$ dont le support appartient à $\mathsf{F}(X)$. Si $u' \in \mathsf{F}(X)^\perp$, l'ensemble des $x \in \mathbf{k}\langle X\rangle$ dont le support ne rencontre pas u' est un sous-espace vectoriel de $\mathbf{k}\langle X\rangle$. Ces sous-espaces forment une base de filtre et définissent donc une topologie linéaire sur $\mathbf{k}\langle X\rangle$, au sens de Lefschetz. L'égalité $\mathsf{F}(X) = \mathsf{F}(X)^{\perp\perp}$ garantit la complétude (au sens de Cauchy) de ce ltvs. Étant donné deux espaces de finitude X et Y, on peut exhiber un isomorphisme linéaire entre $\mathbf{k}\langle X \multimap Y\rangle$ (à voir comme un espace de matrices, infinies en général), et l'espace des fonctions linéaires et continues de $\mathbf{k}\langle X\rangle$ vers $\mathbf{k}\langle Y\rangle$. Cet isomorphisme est un homéomorphisme si on munit cet espace de fonctions de la topologie de la convergence uniforme sur tout sous-espace *linéairement compact* (une notion de compacité introduite dans [Lef42] dans le cadre des ltvs généraux). Il généralise en dimension infinie l'isomorphisme habituel entre matrices et fonctions linéaires, une fois des bases données dans l'espace de départ et l'espace d'arrivée : ici, le rôle des bases est joué par $|X|$ et $|Y|$, qu'on peut voir en effet comme des bases de Schauder de $\mathbf{k}\langle X\rangle$ et $\mathbf{k}\langle Y\rangle$.

Cela permet de voir les espaces de finitude comme des ltvs. Plus précisément, si on définit un *espace de finitude abstrait* comme un $\mathbf{k}$-ltvs qui est linéairement homéomorphe à un $\mathbf{k}\langle X\rangle$, alors, en utilisant les constructions combinatoires sur les espaces de finitude, on peut montrer que les espaces de finitude abstraits (avec les fonctions linéaires continues comme morphismes) forment un modèle de la logique linéaire. Par exemple, la construction $X \otimes Y$ sur les espaces de finitude permet de définir un produit tensoriel $E \otimes F$ sur les espaces de finitude abstraits, dont on peut montrer qu'il classifie les fonction bilinéaires «hypocontinues» (une notion purement algébrico-topologique, plus faible que la continuité) entre tels espaces. De même, le dual topologique E' d'un espace de finitude abstrait, muni de la topologie de la convergence sur tout linéairement compact, est un

espace de finitude abstrait (si E est linéairement homéomorphe à $\mathbf{k}\langle X\rangle$, alors E' est linéairement homéomorphe à $\mathbf{k}\langle X^\perp\rangle$) ; l'application d'un «covecteur» $x' \in \mathsf{F}(X^\perp)$ a un vecteur $x \in \mathbf{k}\langle X\rangle$ est bien définie, par la formule habituelle $\langle x, x'\rangle = \sum_{a\in|X|} x_a x'_a$, puisque cette somme a un nombre fini de termes non nuls.

Les «bonnes» fonctions bilinéaires entre espaces de finitude ne sont donc pas continues en général, mais seulement hypocontinues (c'est un phénomène déjà connu dans les espaces localement convexes). Elles sont continues quand les espaces de départ sont localement linéairement compact. Il est intéressant de noter que cette condition, pour l'espace $\mathbf{k}\langle X\rangle$, équivaut à l'existence de $u \in \mathsf{F}(X)$ et de $u' \in \mathsf{F}(X^\perp)$ tels que $u \cup u' = |X|$. Dans le même ordre d'idée, on montre que $\mathbf{k}\langle X\rangle$ est métrisable si et seulement si on peut trouver une suite croissante $(u'_n)_{n\in\mathbf{N}}$ d'éléments de $\mathsf{F}(X^\perp)$ telle que tout élément de $\mathsf{F}(X^\perp)$ soit inclus dans un u'_n (condition plus faible que celle qui correspond à la compacité linéaire locale). Un argument de diagonalisation à la Cantor montre que cette propriété n'est pas vérifiée par l'espace de finitude !?1 : la logique linéaire fait apparaître des ltvs topologiquement très complexes.

La catégorie $\mathcal{L}_\mathbf{k}$ des espaces de finitude et des fonction linéaires continues constitue donc un modèle de la logique linéaire multiplicative additive MALL (une catégorie $*$-autonome avec sommes et produits, dans le langage de Michael Barr [Bar79]).

Comme dans les espaces de Köthe (et dans la sémantique quantitative de Girard), les exponentielles sont interprétées au moyen de «séries entières». À tout espace de finitude X, on associe un nouvel espace $!X$ en prenant pour $|!X|$ l'ensemble des multi-ensembles finis d'éléments de $|X|$ et en disant qu'une collection de tels multi-ensembles est finitaire si la réunion des supports de ses éléments est finitaire (au sens de X) ; on vérifie que l'ensemble $\mathsf{F}(!X)$ est égal à son bi-orthogonal. Étant donné $x \in \mathbf{k}\langle X\rangle$, on définit $x^! \in \mathbf{k}\langle !X\rangle$ (le *promu* de x, en terme logique) en posant $x^!_m = x^m = \prod_{a\in|X|} x_a^{m(a)}$ (pour chaque $m \in |!X|$, l'entier $m(a)$ est le nombre d'occurrence de a dans le multi-ensemble m vu ici comme multi-exposant). Une matrice $M \in \mathbf{k}\langle !X \multimap Y\rangle$ peut donc être appliquée à un vecteur x de $\mathbf{k}\langle X\rangle$ (de façon non linéaire) pour obtenir un vecteur $M(x) \in \mathbf{k}\langle Y\rangle$, en posant $M(x) = M \cdot x^! = (\sum_{m\in|!X|} M_{m,b} x^m)_{b\in|Y|}$, et M peut être vue comme une série entière de $\mathbf{k}\langle X\rangle$ vers $\mathbf{k}\langle Y\rangle$ (cette somme n'a qu'un nombre fini de termes non nuls, mais ce nombre dépend de x et de b, et n'est pas borné en général).

L'opération «!» est un foncteur de la catégorie $\mathcal{L}_\mathbf{k}$ dans elle-même, qui a une structure de comonade, et la catégorie des espaces de finitude et des séries entières est la *catégorie de Kleisli* de cette comonade. Un morphisme de X vers Y dans cette catégorie peut donc être vu comme une fonction de $\mathbf{k}\langle X\rangle$ dans $\mathbf{k}\langle Y\rangle$, et cette fonction détermine entièrement le morphisme dès que $\mathbf{k}$ est infini (ce qui généralise le fait bien connu qu'un polynôme est complètement déterminé par la fonction associée dès que le corps est infini). On appellera *fonctions entières* ces fonctions dans la suite. Elles ne sont bien sûr pas linéaires, mais elles ne sont en fait pas même continues (de même que les fonctions bilinéaires ne sont pas continues, mais seulement hypocontinues), pour les topologies canoniquement associées à $\mathbf{k}\langle X\rangle$ et $\mathbf{k}\langle Y\rangle$.

Apparition de la structure différentielle des exponentielles. Une série $M \in \mathbf{k}\langle !X \multimap Y\rangle$, ou, ce qui revient au même, une fonction entière $\mathbf{k}\langle X\rangle \to \mathbf{k}\langle Y\rangle$, peut être dérivée. Cela se fait au moyen de deux opérations :

— La «dérivation en 0» qui est calculée par composition avec un morphisme linéaire de X dans $!X$ (une inclusion canonique en fait, que j'ai déjà évoquée en 4.3.4) ; ce faisant, on obtient un élément de $\mathbf{k}\langle X \multimap Y\rangle$ qui est la partie linéaire de M.

— La translation d'une série, qui se calcule au moyen du «produit de convolution». C'est un morphisme linéaire (une matrice dont les coefficients sont des coefficients binomiaux) de $!X \otimes !X$ dans $!X$; par composition avec ce morphisme, on passe de M à $N \in \mathbf{k}\langle !X \otimes !X \multimap Y\rangle$, que l'on peut voir comme une série entière à deux paramètres, et le produit de convolution est tel que l'on a $N(x, y) = M(x + y)$. Il permet donc de translater la série M en n'importe quel point de $\mathbf{k}\langle X\rangle$, et donc, au moyen de la dérivation en 0, de dériver M en n'importe quel point.

La même structure est présente dans les espaces de Köthe, et est à l'origine de l'introduction du lambda-calcul et de la logique linéaire différentiels (voir plus loin).

La rédaction de l'article correspondant ([Ehr05]) aura été plutôt chaotique car j'y avais d'abord intégré une partie sur l'interprétation du second ordre, que j'ai ensuite remplacée par une partie sur le point de vue «espaces vectoriels topologiques» expliqué ci-dessus, si bien que l'article n'a été définitivement accepté qu'en 2004. On reviendra plus tard sur le second ordre.

Fonctions entières comme complétion des polynômes. Les morphismes de la catégorie $\mathcal{L}_\mathbf{k}$ sont les fonctions linéaires continues, et on peut se demander s'il existe aussi une caractérisation plus abstraite des fonctions entières, introduites ci-dessus. J'ai obtenu deux résultat de cette nature.

Étant donné deux ltvs E et F, une fonction $f : E \to F$ est polynomiale si on peut trouver des fonctions multilinéaires hypocontinues $h_i : E^n \to F$ $(i = 0, \ldots, n)$ telles que $f(x) = h_0 + h_1(x) + \cdots + h_n(x, \ldots, x)$. J'ai montré que $\mathbf{k}\langle !X \multimap Y\rangle$ est linéairement homéomorphe à la complétion du ltvs des fonctions polynomiales $\mathbf{k}\langle X\rangle \to \mathbf{k}\langle Y\rangle$, muni de la topologie de la convergence uniforme sur tout linéairement compact.

Fonctions entières comme morphismes de préfaisceaux. La seconde caractérisation est plus globale. Soit $\mathbf{Pol}_\mathbf{k}$ la catégorie des polynômes à coefficients dans $\mathbf{k}$ (un objet de cette catégorie est un entier naturel, et un morphisme de m dans n dans cette catégorie est une n-uplet de polynômes à m indéterminées). Il est standard de plonger cette catégorie dans la catégorie $\mathbf{Set}^{\mathbf{Pol}_\mathbf{k}^{\mathrm{op}}}$ (le topos des préfaisceaux sur $\mathbf{Pol}_\mathbf{k}$) au moyen du *foncteur de Yoneda* $\mathcal{Y}$ qui, à l'objet n de $\mathbf{Pol}_\mathbf{k}$, associe le foncteur contravariant (préfaisceau) $\mathbf{Pol}_\mathbf{k} \to \mathbf{Set}$ qui, à m, associe l'ensemble $\mathbf{Pol}_\mathbf{k}(m, n)$ des n-uples de polynômes à m indéterminées ; le foncteur de Yoneda est plein et fidèle (c'est-à-dire, surjectif et injectif sur les morphismes). Par ailleurs, il est facile de voir que $\mathbf{Pol}_\mathbf{k}$ est une sous-catégorie pleine de $\mathcal{E}_\mathbf{k}$, la catégorie des espaces de finitude et des fonctions entières (il n'y a qu'un seul espace de finitude n dont la trame est $\{1, \ldots, n\}$, et un morphisme de m dans n dans $\mathcal{E}_\mathbf{k}$, autrement dit, une fonction entière de $\mathbf{k}^m \to \mathbf{k}^n$ est tout simplement un n-uple de polynômes à m indéterminées).

J'ai montré que (sous une condition restrictive sur $\mathbf{k}$) le foncteur $\mathcal{Y}$ peut être étendu à toute la catégorie $\mathcal{E}_\mathbf{k}$, en un foncteur $\tilde{\mathcal{Y}}$ plein et fidèle, et que de plus, ce foncteur préserve la structure cartésienne fermée de $\mathcal{E}_\mathbf{k}$. Autrement dit, la catégorie des espaces de finitude et des fonctions entières est une sous catégorie cartésienne fermée pleine du topos des préfaisceaux sur la catégorie des polynômes.

Plus concrètement : on définit un $\mathbf{Pol}_\mathbf{k}$-espace [22] comme un ensemble S de points, muni d'une famille $(S_n)_{n\in\mathbf{N}}$ telle que S_n soit un ensemble de fonctions $\gamma : \mathbf{k}^n \to S$ (intuitivement, les «courbes polynomiales de dimension au plus n» dans S), qui vérifie :

— si $P \in \mathbf{Pol}_\mathbf{k}(m, n)$ et $\gamma \in S_n$, alors $\gamma \circ P \in S_m$ (les «courbes» sont stables par reparamétrisation polynomiale)

22. Ces espaces sont des préfaisceaux particuliers sur $\mathbf{Pol}_\mathbf{k}$, dit *préfaisceaux extensionnels*.

— et le plongement évident de S_0 dans S est surjectif (par tout point passe au moins une «courbe»).

Ces $\mathbf{Pol_k}$-espaces peuvent être organisés en une catégorie : un morphisme de S vers T dans cette catégorie est une fonction $f : S \to T$ telle que, pour tout n et tout $\gamma \in S_n$, on a $f \circ \gamma \in T_n$. Autrement dit, un morphisme de $\mathbf{Pol_k}$-espaces est une fonction qui transforme les courbes polynomiales en courbes polynomiales.

On peut alors voir le foncteur $\tilde{\mathcal{Y}}$ comme associant à tout espace de finitude X un $\mathbf{Pol_k}$-espace S : l'ensemble des points est $\mathbf{k}\langle X\rangle$, et S_n est l'ensemble des fonctions entières de $\mathbf{k}^n = \mathbf{k}\langle n\rangle$ dans $\mathbf{k}\langle X\rangle$. Le résultat que j'ai montré exprime donc que les morphismes de X vers Y, considérés comme $\mathbf{Pol_k}$-espaces, sont exactement les fonctions entières de $\mathbf{k}\langle X\rangle$ vers $\mathbf{k}\langle Y\rangle$.

Pour l'instant, ce résultat est limité au cas où $\mathbf{k}$ est infini, bien sûr (pour pouvoir voir les morphismes de la catégorie de Kleisli de «!» comme des fonctions), mais je fais en plus l'hypothèse que $\mathbf{k}$ admet une norme pour laquelle il est complet, et non discret — cette norme n'est pas utile pour la description de la catégorie des $\mathbf{Pol_k}$-espaces, mais uniquement pour la preuve de plénitude de $\tilde{\mathcal{Y}}$, car j'utilise le théorème de catégoricité de Baire dans la dernière étape de cette preuve (les réels, les complexes, les p-adiques vérifient cette condition). Ce résultat est présenté dans le rapport [Ehr07], et dans un papier [Ehr09] qui a été accepté pour publication dans *Journal of Pure and Applied Algebra* (il est en cours de révision depuis trop longtemps, faute de trouver le temps d'y travailler sérieusement).

4.4.3 Espaces «convenient» (depuis 2008)

Dans le but d'obtenir des modèles de la logique linéaire à base d'espaces vectoriels topologiques et qui soient intrinsèques (au sens où ils ne font pas intervenir de choix privilégiés de bases, comme les espaces de finitude ou les espaces de Köthe), nous nous sommes intéressés, avec Blute (université d'Ottawa) et Tasson à la notion de «convenient space», introduite par Frölicher dans les années 80. Ce sont des espaces vectoriels topologiques dont la topologie est déterminée par la bornologie [23]. On sait depuis les travaux de Fröhlicher que ces espaces, avec une notion adéquate de fonction C^∞, forment une catégorie cartésienne fermée. Nous avons montré que cette CCC dérive d'un modèle de la logique linéaire différentielle, introduisant une construction exponentielle sur les espaces convénients.

Les espaces convénients possèdent une propriété de complétude (au sens de Cauchy) plus faible que la notion topologique habituelle, et qui est définie en terme de parties bornées : c'est la Mackey-complétude. Celle-ci est toutefois suffisante pour pouvoir parler de dérivées itérés, en s'inspirant de la notion de fonction Lipschitzienne, qui est bien un concept «bornologique».

Ce travail fait l'objet de [BET12], qui a été accepté et va paraître prochainement.

4.5 Lambda-calcul et logique linéaire différentiels (depuis 2003)

Dans le modèle des espaces de finitude — et la situation est similaire dans tous les modèles dénotationnels de la logique linéaire — un lambda-terme de type $A \to B$ est interprété par un élément de $\mathbf{k}\langle !A \multimap B\rangle$ (en notant de la même façon un type et l'espace de finitude qui l'interprète), c'est-à-dire ici par une série entière. Cette interprétation peut donc être dérivée, ce qui produit un

23. Dans tout espace vectoriel topologique E, il y a une notion de partie bornée : ce sont les parties B telles que, pour tout voisinage U de 0, il existe un $\lambda > 0$ tel que $B \subseteq \lambda U$ (on dit alors que B est absorbée par U). L'ensemble des parties bornées de E forme la bornologie de E. Contrairement à ce qui se passe dans les espaces de Banach, il n'y a généralement pas de voisinage borné de 0 ; topologie et bornologie sont des concepts distincts et duaux.

élément de $\mathbf{k}\langle !A \multimap (A \multimap B)\rangle$ (une série entière à valeurs dans un espace de fonctions linéaires). Il semblait donc naturel de chercher un analogue syntaxique de cette opération en introduisant dans le lambda-calcul une *dérivation formelle*.

4.5.1 Lambda-calcul différentiel

Avec Regnier, j'ai introduit le lambda-calcul différentiel dans [ER03]. Cela a nécessité d'étendre la syntaxe du lambda-calcul dans deux directions :

— D'une part, il a fallu permettre d'additionner des termes de même type. En effet, si l'on dérive un terme t par rapport à une variable x, et que cette variable a plusieurs occurrences dans t, on obtient une somme de plusieurs termes obtenus en prenant des dérivées partielles correspondant à ces différentes occurrences.

— D'autre part, il a fallu introduire un opérateur de dérivation (ou de différentiation) dans la syntaxe. Pour éviter d'avoir à toucher au système de types, nous avons fait le choix suivant qui s'est révélé commode dans la pratique. Étant donné un terme t de type $B = A_1 \to (\cdots \to (A_n \to A)\cdots)$ et un terme u de type A_n, nous introduisons le terme $\mathrm{D}_n t \cdot u$ de type B. L'intuition est la suivante : t est considéré comme une fonction à n paramètres, donc la différentielle de t par rapport à son n-ième paramètre devrait être une fonction t' de type $A_1 \to (\cdots \to (A_n \to (A_n \multimap A))\cdots)$. Le terme $\mathrm{D}_n t \cdot u$ représente cette fonction, à laquelle on a fourni son $(n+1)$-ième paramètre, u, de type A_n.

Sans en changer la nature, la présentation cette opération de dérivation peut être simplifiée : les indices sont inutiles, comme l'a montré Lionel Vaux dans [Vau05].

Nous avons défini, par récurrence sur le terme t, sa *dérivée partielle* par rapport à une variable x, appliquée (linéairement) à un terme u, ce que nous notons $\frac{\partial t}{\partial x} \cdot u$. Pour donner une idée de cette définition inductive, voici le cas où t est une application, $t = (t_1)\, t_2$ (en lambda-calcul, il est commode de mettre les parenthèses autour de la fonction et non autour de l'argument comme on le fait d'habitude) :

$$\frac{\partial (t_1)\, t_2}{\partial x} \cdot u = \left(\frac{\partial t_1}{\partial x} \cdot u\right) t_2 + \left(\mathrm{D}_1 t_1 \cdot \left(\frac{\partial t_2}{\partial x} \cdot u\right)\right) t_2 \,.$$

La somme qui apparaît dans cette formule correspond au fait que la variable x peut être libre à la fois dans t_1 et dans t_2. Le premier terme de cette somme reflète le fait que l'application est une opération qui est linéaire en la fonction (et la dérivation commute aux opérations linéaires). Le second terme correspond à la «règle de la chaîne» du calcul différentiel ordinaire (dérivation d'une composition de fonctions).

Si t est une variable y, $\frac{\partial t}{\partial x} \cdot u$ vaut u si $x = y$ et 0 sinon.

Pour finir de donner une idée de ce système, il faut préciser les règles de réduction.

— On retrouve la β-réduction ordinaire : $u = (\lambda x\, s)\, t$ se réduit en $s[t/x]$ (s dans lequel on remplace partout la variable x par le terme t).

— Et on introduit la *β-réduction différentielle* : le terme $v = \mathrm{D}_1(\lambda x\, s) \cdot u$ se réduit en $\lambda x\, (\frac{\partial s}{\partial x} \cdot u)$.

Un terme de la forme u ou v est appelé un *redex*, et on dit qu'un terme t se réduit en un terme t' en une étape si on peut passer de t à t' en choisissant un redex dans t et en le réduisant. On dit que t est *normal* s'il ne contient pas de redex.

Nous avons montré tout d'abord que cette réduction est confluente [24]. Comme d'habitude, cela assure qu'un terme normalisable a exactement une forme normale. Nous avons également prouvé, dans le cas typé (du premier et du second ordre), que tout terme de ce système est fortement normalisable [25]. C'est un théorème essentiel qui signifie que, considérées d'un point de vue logique, les règles différentielles que nous avons ajoutées à la logique intuitionniste (lambda-calcul typé) ne remettent pas en cause la propriété cruciale d'élimination des coupures. Cette preuve se fait en adaptant la méthode de réductibilité (Tait pour le premier ordre, étendue par Girard au second ordre).

Nous avons voulu aller plus loin en cherchant à comprendre le lien entre application différentielle $\mathrm{D}_1 s \cdot t$ et application ordinaire $(s)\, t$ d'une part, et substitution différentielle *alias* dérivée partielle $\frac{\partial s}{\partial x} \cdot t$, et substitution ordinaire $s[t/x]$ de l'autre. On sait que les deux notions doivent être reliées par la formule de Taylor (valide dans les modèles), qui énonce que

$$(s)\, t = \sum_{n=0}^{\infty} \frac{1}{n!} \left(\mathrm{D}_1^n s \cdot \overbrace{(t, \ldots, t)}^{n\times} \right) 0 \ . \tag{1}$$

Dans cette formule, l'expression $\mathrm{D}_1^n s \cdot (t_1, \ldots, t_n)$ représente le terme différentiel $\mathrm{D}_1(\cdots \mathrm{D}_1 s \cdot t_1 \cdots) \cdot t_n$. Nous avons prouvé le résultat suivant : si s et t sont des lambda-termes ordinaires (sans application différentielle), et si le terme $(s)\, t$ est β-équivalent à une variable distinguée $\star$, alors il existe exactement un entier n tel que le lambda-terme différentiel $(\mathrm{D}_1^n s \cdot (t, \ldots, t))\, 0$ ne soit pas égal à 0 (pour l'égalité définie par les règles de réduction que nous avons données ci-dessus), et que pour cet entier n, le terme $(\mathrm{D}_1^n s \cdot (t, \ldots, t))\, 0$ est égal à $n!\star$. Cette valeur n est le nombre de fois où le terme t arrive en tête au cours de la *réduction linéaire de tête* de $(s)\, t$ à sa forme normale $\star$.

Cette notion de réduction sur le lambda-calcul, distincte de la beta-réduction (bien qu'elle lui soit fondamentalement équivalente), est celle qui est implémentée par la Machine de Krivine, une machine à environnement qui est à l'origine de l'implémentation actuelle du langage Ocaml. C'est une implémentation réaliste du lambda-calcul au sens où, lors de la substitution, le substituant n'est pas copié : on effectue la substitution uniquement sur l'occurrence de tête (la seule occurrence linéaire) de la variable de tête du substitué. Cela est rendu possible par la présence de l'environnement. L'exécution se ramène à la gestion d'un pointeur qui se déplace dans le terme à exécuter, grâce à une structure de donnée qui est l'état courant de la machine.

4.5.2 Lambda-calcul avec ressources, formule de Taylor et machine de Krivine

Cette analyse de la formule de Taylor montre que l'application ordinaire du lambda-calcul peut laisser la place à une notion plus primitive d'application, multi-linéaire celle-là. Cela mène naturellement à une nouvelle syntaxe, dans laquelle on peut appliquer un terme s à un *multi-ensemble* fini T d'arguments (ces arguments sont des termes $t_1, \ldots, t_n$, et T est le multi-ensemble $T = [t_1, \ldots, t_n]$), ce qu'on a choisi de noter $\langle s \rangle\, T$ pour éviter toute confusion avec l'application ordinaire.

Dans le lambda-calcul différentiel, le terme correspondant s'écrirait $(\mathrm{D}_1^n s \cdot (t_1, \ldots, t_n))\, 0$ (dans cette expression, l'ordre entre les arguments n'importe pas, c'est pourquoi nous pouvons utiliser des multi-ensembles dans la nouvelle syntaxe). Le *poly-terme* T peut être vu comme un produit

24. Si un terme t se réécrit (en plusieurs étapes) en t_1 et aussi en t_2, alors il existe un terme t' tel que t_1 et t_2 se réécrivent (en plusieurs étapes) en t'.

25. C'est-à-dire qu'il n'y a pas de suite de termes $(t_i)_{i \in \mathbf{N}}$ telle que t_i se réécrive en t_{i+1} en une étape pour chaque i.

des termes $t_1, \ldots, t_n$ et la sémantique l'interprète effectivement comme le produit de convolution [26] (qui est commutatif) de l'interprétation de ces termes ; on écrira $T = t_1 \ldots t_n$ et on dénotera par 1 le poly-terme vide. Dans ce calcul, il n'y a plus de β-réduction, mais seulement une β-réduction différentielle :

— $\langle \lambda x\, s \rangle\, 1$ se réduit en $s[0/x]$ (ce terme valant 0 si x apparaît libre dans s et s sinon

— et $\langle \lambda x\, s \rangle\, tT$ se réduit en $\left\langle \lambda x \left(\frac{\partial s}{\partial x} \cdot t \right) \right\rangle T$.

La dérivation partielle se définit par récurrence sur les termes, comme dans le lambda-calcul différentiel. La confluence se prouve comme dans le cas précédent, et le calcul est fortement normalisant (même dans le cas non typé), car une mesure simple de la taille des termes diminue strictement au cours de la réduction.

Ce calcul, dont l'origine est la découverte de structures différentielles dans certains modèles dénotationnels de la logique linéaire et du lambda-calcul, retrouve des idées antérieures de Gérard Boudol [Bou93, BCL99], qui a mis au point un lambda-calcul avec ressources avec des motivations très différentes : le but était de mieux comprendre la traduction du lambda-calcul dans le pi-calcul de Milner.

Développement de Taylor itéré d'un lambda-terme. Tout lambda-terme ordinaire peut être traduit en une somme formelle infinie de termes de notre calcul avec ressources (en prenant le développement de Taylor de chacune des applications qui y apparaissent et en appliquant systématiquement la formule du multinôme), avec des coefficients, inverses d'entiers, qui reflètent les répétitions de sous-termes (tout comme le $n!$ du développement de Taylor «à un étage») : c'est le développement de Taylor du lambda-terme. Plus précisément, on peut linéariser un terme avec ressources (en rendant distinctes les occurrences de variables, tout en se souvenant de quelle variable linéarisée correspond à quelle variable du terme de départ), et alors l'inverse de ce coefficients est l'ordre du groupe des permutations de variables laissant le terme linéarisé invariant (en tenant compte du fait que les multi-ensembles sont commutatifs).

Les termes avec ressources obtenus en développant les lambda-termes ordinaires sont très particuliers : tous les éléments d'un multi-ensemble donné dans un tel terme ont «la même allure» puisqu'ils viennent tous d'un même lambda-terme. Par exemple, le développement de $(x)(x)x$ (x appliqué à l'application de x à x) contient les termes $\langle x \rangle\, \langle x \rangle\, 1$ (avec coefficient 1) et $\langle x \rangle\, (\langle x \rangle\, x^2)(\langle x \rangle\, x^3)^2$ (avec coefficient $1/(2 \times 6^2) = 1/72$), parmi une infinité d'autres. Ces termes avec ressources particuliers sont cohérents entre eux (pour une relation binaire de cohérence qui est facile à définir en toute généralité sur les termes avec ressources), et en particulier cohérents avec eux-mêmes (on dira «uniformes»).

Propriétés de la réduction du développement de Taylor : cas uniforme. Cette notion de cohérence et d'uniformité a de très bonnes propriétés vis-à-vis de la notion de réduction définie sur les termes avec ressources, et cela permet de montrer que, si l'on prend le développement de Taylor d'un lambda-terme ordinaire et que l'on mène à sa forme normale chacun des termes avec ressources qui y figurent, alors :

26. Plus précisément – dans le modèle des espaces de finitude –, si X est le type des t_i, on a une inclusion de $\mathbf{k}\langle X \rangle$ dans $\mathbf{k}\langle !X \rangle$, espace qui peut être vu comme un espace de distributions sur $\mathbf{k}\langle X \rangle$, puisque c'est le dual topologique de $\mathbf{k}\langle (!X)^\perp \rangle$ et que les éléments de $\mathbf{k}\langle (!X)^\perp \rangle$ sont des séries entières de $\mathbf{k}\langle X \rangle$ dans $\mathbf{k}$. L'inclusion de $\mathbf{k}\langle X \rangle$ dans $\mathbf{k}\langle !X \rangle$ correspond à l'opération qui prend un vecteur u et l'envoie sur la distribution qui prend la dérivée en 0 dans la direction de u. Le produit des t_i correspond au produit de convolution de ces dérivées directionnelles pour chacun des t_i.

— dans les formes normales de deux termes distincts de ce développement, il n'y a pas de terme avec ressource normal commun, en particulier, la somme infinie de ces formes normales conserve un sens à travers la réduction (pas évident *a priori* !) ;
— tout se passe bien avec les coefficients, si bien que la forme normale du développement de Taylor est le développement de Taylor de l'arbre de Böhm (une notion infinitaire de forme normale, pour des lambda-termes qui ne sont pas forcément normalisables) du lambda-terme en question.

En particulier, pour tout terme t avec ressource (forcément normal) figurant dans le développement de Taylor de l'arbre de Böhm d'un lambda-terme ordinaire M, on peut trouver exactement un terme t_0 du développement de Taylor de M tel que t apparaisse dans la forme normale de t_0.

Ces propriétés généralisent bien ce qui avait été observé dans le premier articles sur le lambda-calcul différentiel.

Une première partie de ces résultats fait l'objet d'un premier rapport [ER05], dont la version corrigée a paru dans *Theoretical Computer Science* [ER08]. Le lien avec les arbres de Böhm (et aussi la machine de Krivine) est exposé dans [ER06a] (extended abstract), dont il existe aussi une version plus longue [ER06b].

Propriétés de la réduction du développement de Taylor : cas non uniforme (depuis 2009). La preuve de ces deux propriétés de la réduction vis-à-vis du développement de Taylor utilise crucialement l'uniformité. C'est particulièrement frappant pour la deuxième, qui met en oeuvre des constructions sur les groupes de permutation de variables qui ne semblent possibles que dans le cas uniforme.

J'ai donc cherché à poursuivre cette analyse dans le cadre non-uniforme, en renonçant à la relation de cohérence sur les termes avec ressources. Je l'ai remplacée par une notion de finitude au sens des espaces de finitude 4.4.2. C'est en effet le cadre non uniforme le plus simple que je connaisse et qui permette de contrôler les coefficients au cours de la réduction. Ce contrôle est obtenu comme suit :

— on définit une relation d'ordre sur les termes avec ressources : $s \leq t$ si s apparaît dans un réduit de t (ie. un réduit de t est une somme dont s est l'un des constituants simples)
— et on dit qu'un ensemble u de termes simples est finitaire s'il a une intersection finie avec tout cône $\uparrow t = \{s \mid t \leq s\}$.

Par construction, cela définit un espace de finitude $\mathcal{N}$ sur les termes avec ressources simples. Par construction également, pour n'importe quelle combinaison linéaire finitaire x de termes avec ressources ($x \in \mathbf{k}\langle\mathcal{N}\rangle$ où $\mathbf{k}$ est un corps quelconque), on peut réduire les différents constituants simples de x sans jamais constituer d'accumulation non bornée sur un terme simple. En particulier on peut définir une fonction de normalisation $\mathbf{k}\langle\mathcal{N}\rangle \to \mathbf{k}\langle\mathcal{N}\rangle$ qui est une projection continue (on rappelle que $\mathbf{k}\langle\mathcal{N}\rangle$ a une structure naturelle d'espace vectoriel à topologie linéaire).

On peut étendre le lambda-calcul pur en autorisant la formation de combinaisons linéaires finies à coefficients dans $\mathbf{k}$: la beta-réduction a encore un sens dans ce cadre, et le développement de Taylor de tels termes mène à des combinaisons linéaires de termes avec ressources simples qui ne sont pas uniformes (et ont des supports qui ne sont pas des cliques au sens du paragraphe précédent). Ces supports ne sont pas non plus finitaires au sens de $\mathcal{N}$, comme on s'en rend compte en considérant un lambda-terme comme $M = (\Theta)\lambda\, x(x + z)$ où Θ est l'opérateur de point fixe de Turing : la réduction de ce terme mène successivement à $z + M$, $2z + M$, $3z + M \ldots$ En utilisant une technique de réductibilité (à la Girard, dans la présentation très élégante donnée par Krivine), on montre

que ces lambda-termes ont des développements de Taylor finitaires dès qu'ils sont typables dans le système F (ou plutôt une extension algébrique évidente du système F). Ce résultat est obtenu en associant à chaque type de F un espace de finitude dont la trame est un ensemble de termes avec ressources simples.

En utilisant le fait qu'un terme avec ressources n'a qu'un nombre fini de minorants pour la relation d'ordre définie ci-dessus sur les termes simples, on montre que le ltvs $\mathbf{k}\langle\mathcal{N}\rangle$ (qui interprète les types atomiques du système F) est métrisable et donc que sa topologie est relativement simple. Cette propriété est sûrement perdue pour les ltvs correspondant aux types plus complexes, et il serait intéressant d'analyser cette complexité topologique, peut-être avec des outils de théorie descriptive des ensembles, et de la relier à la beta-réduction et au rôle du système F dans la preuve de cohérence relative de l'arithmétique du second ordre.

Ce travail a fait l'objet de [Ehr10], accepté à LICS en 2010. Ce n'est qu'une étape préliminaire d'un programme visant à remplacer le lambda-calcul par un formalisme plus élémentaire, le lambda-calcul avec ressources, et à comprendre les structures mathématiques pertinentes dont on peut équiper ces termes atomiques. Une autre partie de ce programme consiste à analyser le comportement des coefficients du développement de Taylor itéré au cours de la réduction. Dans le cas uniforme, on a obtenu des résultats très satisfaisants en étudiant le groupe des permutations d'occurrences de variables laissant un terme invariant. Il s'agirait de mener une étude similaire dans le cadre non uniforme, où des superpositions peuvent se produire, ce qui était impossibles dans le cas uniforme.

Le travail en cours avec Pagani et Tasson mentionné en 4.9 vise également à développer un analogue probabiliste ce cette théorie.

4.5.3 Logique linéaire différentielle

Il était clair depuis le début de notre travail sur le lambda-calcul différentiel (ce point est mentionné dans l'introduction de [ER03]) qu'une extension différentielle de la logique linéaire s'imposait tout autant que celle que nous avons approfondie dans un premier temps, pour le lambda-calcul. Il faut d'ailleurs mentionner que Girard, dans la conclusion de [Gir87], indique avoir considéré la possibilité d'introduire des notions différentielles — dont la nature n'est pas très explicite — dans la logique linéaire. Il ne l'a pas fait, peut-être parce que l'extension différentielle de la logique linéaire est incompatible avec la sémantique des espaces cohérents, et aussi avec la totalité.

Une nouvelle symétrie. Le premier système de règles logiques et de règles de réduction (élimination des coupures) auquel nous avons pensé était basé sur le morphisme de type $!X \otimes X \multimap !X$ qui permet d'obtenir directement, par pré-composition, la différentielle d'un morphisme de type $!X \multimap Y$ (correspondant à une fonction entière de X dans Y, à une preuve intuitionniste de Y à partir de X ou à un programme fonctionnel de type $X \to Y$, selon le point de vue). On obtient alors un système de réseaux de preuve (une extension des réseaux de la logique linéaire) qui fonctionne bien mais n'a pas de propriétés géométriques particulièrement remarquables.

Cependant, si l'on fait un autre choix de morphismes pour présenter la logique linéaire différentielle, la situation est toute autre. Dans ce qui suit, $f : !X \multimap Y$ correspond à une fonction entière g de X dans Y. On choisit

- le morphisme $1 \multimap !X$ de *coaffaiblissement* : en le composant avec f on obtient $g(0)$ dans Y ;
- le morphisme $!X \otimes !X \multimap !X$ de *co-contraction* : en le composant avec f on obtient la fonction entière h avec deux paramètres dans X, $h(x_1, x_2) = g(x_1 + x_2)$ (on a appelé aussi cette co-contraction «produit de convolution» dans ce qui précède) ;

— le morphisme $X \multimap !X$ de *codéréliction* : en le composant avec f on obtient une fonction linéaire de X dans Y, la différentielle de g en 0.

On a alors, du moins formellement, une symétrie remarquable avec les morphismes suivants qui correspondent à des principes de base de la logique linéaire ordinaire :

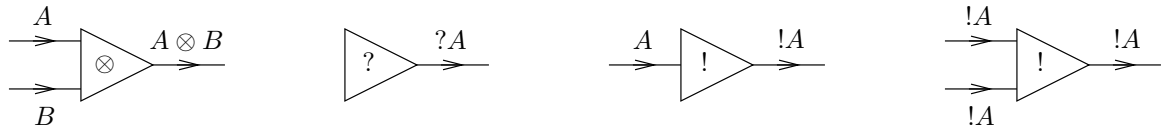
— le morphisme $!X \multimap 1$ d'*affaiblissement* : à partir d'une valeur y dans Y (vue comme un morphisme $1 \multimap Y$), il permet par composition de construire la fonction entière constante sur X, de valeur y (logiquement : permet de ne pas utiliser une hypothèse) ;

— le morphisme $!X \multimap !X \otimes !X$ de *contraction* : en le composant avec une fonction entière h à deux paramètres dans X, on obtient la fonction g à un seul paramètre donnée par $g(x) = h(x, x)$ (logiquement : permet d'utiliser deux fois une hypothèse) ;

— le morphisme $!X \multimap X$ de *déréliction* : en le composant avec une fonction linéaire $X \multimap Y$, on obtient une fonction entière (qui n'est autre que la même fonction, dont on a oublié qu'elle a cette propriété de linéarité).

Cette symétrie s'étend aux règles de réduction (élimination des coupures). On peut les représenter dans un style catégorique, par des diagrammes commutatifs, ou par des réseaux de preuve étendant ceux de [Gir87], mais le formalisme des *réseaux d'interaction* (Yves Lafont) est plus approprié [27]. Dans ce cadre, une preuve est représentée par une collection de *cellules*. Chaque cellule possède des *ports* (dont un, toujours présent, est dit *port principal*), et ces ports sont reliés entre eux par des *fils*.

Une cellule représente une règle logique, ou encore, un morphisme, selon le point de vue. Par exemple, voici les cellules correspondant aux règles tenseur, affaiblissement, codéréliction et co-contraction (pour chaque cellule, le port principal est le seul qui se trouve sur l'un des sommets du triangle).



Les fils *orientés* peuvent être typés par des formules de la logique linéaire comme on l'a fait ici [28].

Symétrie de la réduction. Un réseau constitué de telles cellules peut contenir des *redex*, qui sont des configurations consistant en deux cellules reliées par leur ports principaux. À chaque tel redex [29] on associe un réduit qui consiste en un réseau possédant la même interface (fils pendants).

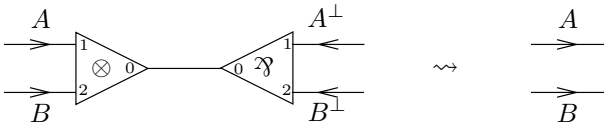
On va maintenant énumérer les redex et montrer leur réduction, pour illustrer la parfaite symétrie de la situation.

On commence par le redex multiplicatif, qui met en jeu une cellule tenseur et une cellule cotenseur (par) :

27. En particulier parce qu'il ne traite pas l'axiome et la coupure de la même façon que les règles logiques : alors que ces dernières donnent lieu à des cellules, les premières sont représentées par de simples fils. Tout comme les réseaux de preuves de la logique linéaire, il a aussi l'avantage de permettre de garder implicites les isomorphismes de monoïdalité de la catégorie sous-jacente.

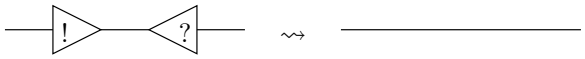
28. Quand on change l'orientation d'un fil typé par A, son type devient $A^{\perp}$, la négation linéaire de A ; rappelons que cette négation est involutive.

29. À condition toutefois qu'il soit typé : deux cellules «tenseur» reliées entre elles par leurs ports principaux ne constituent pas un redex.



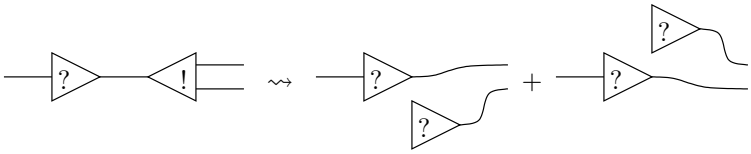
Noter les numéros de ports : 0 pour le port principal, 1 et 2 pour les ports auxiliaires de chaque cellule. On ne les indiquera plus. Bien sûr, pour un tenseur ou un par, il est essentiel de ne pas confondre les ports auxiliaires ($A \otimes B$ et $B \otimes A$ ne sont pas la même formule, même si elles sont équivalentes !).

Voici maintenant les réductions concernant les exponentielles. Le redex le plus simple est le redex de *communication*, qui met en jeu une codéréliction et une déréliction.

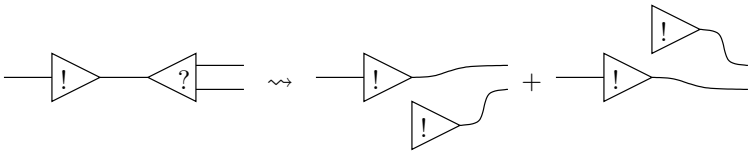


Il exprime que la dérivée en 0 d'une fonction linéaire (vue comme fonction entière au moyen de la déréliction) est cette fonction elle-même.

Comme on l'a vu, le lambda-calcul différentiel impose l'introduction de sommes de termes, à cause du caractère «non-déterministe» de la dérivation (substitution linéaire). Il en va de même ici, et les sommes sont introduites par deux règles de réduction duales l'une de l'autre : la configuration co-contraction/déréliction et la configuration contraction/codéréliction [30], dits redex non-déterministes. Un principe général de distributivité s'applique, si bien qu'on n'a à considérer que des sommes (ou combinaisons linéaires) de réseaux «simples» (sans sommes). Voici donc la première de ces configurations, et sa réduction :



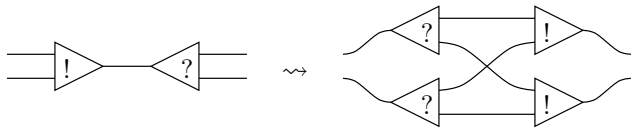
Si l'on revient à la signification «fonctionnelle» de la déréliction donnée page 25 (cellule de gauche dans le redex) et de la co-contraction donnée de même page 24 (cellule de droite), on réalise que cette règle correspond simplement au fait qu'une fonction linéaire appliquée à une somme commute à cette somme. La seconde configuration est



et correspond au cas de la dérivation en 0 (codéréliction, cellule de gauche dans le redex) par rapport à x d'une fonction $f(x, x)$ (l'identification des paramètres de la fonction est réalisée par la contraction, cellule de droite du redex) ; le résultat est $f'_1(0,0) + f'_2(0,0)$, somme des deux dérivées partielles calculées en 0, ce qui correspond très précisément au réduit.

30. De même, les cas affaiblissement/codéréliction et coaffaiblissement/déréliction font apparaître des 0.

Enfin, les redex *structurels* correspondent à la configuration co-contraction/contraction et co-contraction/affaiblissement (et sa duale) ; ce sont des règles de réduction correspondant à l'axiomatique usuelle des bialgèbres [31]. On donne juste ici le cas co-contraction/contraction :



Cette symétrie, tout-à-fait nouvelle en logique linéaire, entre le «!» et le «?» étend aux exponentielles la symétrie déjà connue entre le «⊗» et le «⅋» (par ou cotenseur), son connecteur dual.

On a étudié dans [ER04] les propriétés de base de ces réseaux ; on a d'abord introduit un système de typage avec types «récursifs» qui permet d'éviter l'apparition de configurations non réductibles (comme deux cellules tenseur connectées entre elles par leurs ports principaux) sans introduire trop de contraintes par ailleurs [32]. Puis on a défini un *critère de correction* qui étend à ce cadre le critère de Danos et Regnier pour les réseaux multiplicatifs. Il s'agit d'un critère géométrique global qui, lorsqu'il est satisfait, assure qu'aucun cercle vicieux n'apparaîtra durant la réduction et que, pour cette raison, la réduction terminera toujours (même avec notre système de type très faible qui, dans la logique linéaire, n'assurerait pas la terminaison : tout comme le lambda-calcul avec ressources, les réseaux d'interaction différentiels sont finitaires car ils ne possèdent pas de règle de promotion [33]. Celle-ci peut bien sûr être ajoutée, et on obtient alors une version des réseaux différentiels correspondant au lambda-calcul différentiel présenté en 4.5.1. Nous avons prouvé ce résultat de terminaison général, ainsi que la confluence (en un sens très fort, grâce au caractère local de la réduction).

On peut aussi prouver un théorème de séquentialisation disant qu'un réseau satisfaisant le critère de correction peut être obtenu à partir d'une preuve dans une version *calcul des séquents* de la logique linéaire différentielle, comme c'est le cas pour la logique linéaire multiplicative. Ce calcul des séquents (ou du moins sa version purement exponentielle, mais l'extension aux multiplicatifs ne pose aucun problème) est présenté dans [EL06a, EL06b]. J'ai écrit le papier [Ehr11] présentant plus en détail la logique linéaire différentielle (promotion comprise), son calcul des séquents, ses réseaux et ses modèles, en insistant sur les aspects sémantiques. Ce papier contient aussi des contributions nouvelles sur lesquelles on reviendra (voir 4.8.3).

31. Il faudrait une règle exprimant la neutralité de l'affaiblissement pour la contraction, et dualement ; ces règles ne sont pas au format standard des réseaux d'interaction, où un redex doit être constitué de deux cellules connectées par leurs ports principaux, mais peuvent être ajoutées sans problème comme on le montre dans [EL06b]. Quant à l'associativité et l'éventuelle commutativité de la co-contraction et/ou de la contraction, elle peut être axiomatisée comme une relation d'équivalence sur les réseaux.

32. D'ordinaire, l'introduction des types a pour effet de forcer des propriétés de normalisation, mais ce ne saurait être le cas ici car nos types sont récursifs, c'est-à-dire qu'on se permet d'introduire par exemple un type o vérifiant $o = !o \multimap o = ?(o^{\perp}) \parr o$.

33. En lambda-calcul ordinaire, quand on applique un terme M de type $A \to B = !A \multimap B$ à un terme N de type A, il faut faire subir à ce dernier une promotion, ce qui produit un argument de type $!A$ auquel peut être appliqué M, vu comme fonction linéaire. Cette opération consiste à rendre N indéfiniment copiable par M, «inusable» en quelque sorte. Ce n'est que grâce à cela qu'il est possible d'écrire des lambda-termes dont la réduction ne termine pas, comme le fameux $(\lambda x\,(x)x)\lambda x\,(x)x$ (les parenthèses sont autour des fonctions).

4.5.4 Représentation des calculs de processus

Les calculs de processus, développés par des informaticiens théoriciens depuis les années 80 (à la suite notamment de Milner à qui on doit le pi-calcul), cherchent à donner des bases conceptuelles solides à la programmation parallèle et concurrente [34], quelque chose qui soit à ce style de programmation ce que le lambda-calcul est à la programmation fonctionnelle. Même si la théorie du pi-calcul est très développée (voir par exemple [SW01]), et si des résultat ont été obtenus dans le domaine de la sémantique catégorique des calculs de processus (par Winskel notamment), ces formalismes sont très loin d'avoir une théorie mathématique aussi satisfaisante que celle du lambda-calcul ou de la logique linéaire. On peut attribuer ce déficit à l'absence de correspondance de Curry-Howard entre ces calculs et des systèmes logiques acceptables (jouissant de l'élimination des coupures, possédant une sémantique dénotationnelle etc.).

Or le domaine d'application des calculs de processus s'étend. On commence à les utiliser en biologie pour représenter et même pour modéliser des réseaux d'interaction entre espèces chimiques (protéines typiquement) au sein des cellules. Ils sont par ailleurs utilisés depuis longtemps pour l'étude des protocoles cryptographiques et bien sûr pour la certification des programmes concurrents. En un mot, les calculs de processus semblent entretenir avec le «monde réel» des liens plus forts que le lambda-calcul, ce qui n'est pas très surprenant : contrairement au lambda-calcul (qui vient de la logique), les calculs de processus ont été conçus dans le but de modéliser les phénomènes en question. Une de leurs origines est le langage CCS (Milner) qui vient de la théorie des automates finis : rien à voir donc avec Curry-Howard a priori !

Après la découverte de la logique linéaire, un certain nombres d'informaticiens théoriciens ont pensé que ce nouveau formalisme donnerait une interprétation logique des calculs parallèles et concurrent, à cause de la nature très asynchrone de la réduction dans les réseaux de preuve. Mais cette idée se heurtait au caractère déterministe de l'élimination des coupures de la logique linéaire, alors que les calculs concurrents sont fondamentalement non déterministes. La logique linéaire différentielle semble permettre de surmonter cet obstacle.

Le pi-calcul. Dans le pi-calcul, un processus consiste en une *mise en parallèle* d'une famille finie de processus *gardés*, c'est à dire, de processus qui consistent en un *préfixe* de *réception* ou d'*émission* suivi d'un autre processus (qui lui-même peut être une mise en parallèle de processus gardés etc.). La mise en parallèle est fondamentalement associative et commutative : dans une mise en parallèle $P_1 \mid P_2 \mid \cdots \mid P_n$ (la barre verticale dénote la mise en parallèle), tous les processus gardés P_i ont le même statut les uns vis-à-vis des autres.

Un préfixe d'émission comporte un *nom* [35] dit *sujet* qu'il faut comprendre comme le nom d'un canal, sur lequel une liste d'autres noms, dits *objets* — spécifiée également dans le préfixe —, sera passée par le préfixe lorsqu'il interagira avec une préfixe de réception dont le sujet est le même nom.

Un préfixe de réception comporte également un nom de canal sujet, ainsi qu'une liste de noms objets. Ces derniers ne sont pas destinés à être transmis à l'extérieur : tout comme des variables abstraites par une lambda-abstraction dans un lambda-terme, ils seront substitués, dans le processus qui suit le préfixe, par les noms reçus du préfixe d'émission lors de l'interaction avec celui-ci. Dans un processus gardé par un préfixe de réception, les noms objets sont donc *liés* (ce sont des variables muettes), et ne sont pas visibles de l'extérieur du processus.

34. Au sens où les processus sont en concurrence pour accéder à des ressources.

35. Les noms sont simplement des symboles élémentaires destinés à être substitués, exactement comme les variables du lambda-calcul ou des polynômes. Ici, les noms seront substitués par d'autres noms seulement.

La réduction d'un processus général (multi-ensemble de processus gardés) est donc une opération très naturelle, qui se décompose en deux étapes.

— On choisit dans ce multi-ensemble un processus émetteur et un processus récepteur qui ont le même sujet. Cette étape est non-déterministe.

— Puis on les fait interagir, c'est à dire qu'on supprime les préfixes des deux processus, et on remplace dans le second les occurrences des noms objets du préfixe de réception par les noms objets du préfixe d'émission [36]. Cette étape est déterministe.

Un dernière opération dite de *restriction* permet de rendre un nom local à un processus, interdisant toute communication du processus sur ce nom avec l'extérieur.

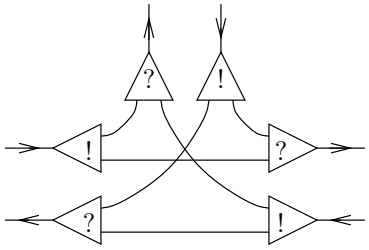
La réduction du pi-calcul est donc fondamentalement non-déterministe.

Le pi-calcul possède d'autres opérations, comme la récursion (qui ne pose pas trop de problèmes vis-à-vis de l'interprétation que nous allons esquisser) et la somme non déterministe, dont l'origine est à trouver dans la théorie des automates (à travers CCS) et qui est plus difficile à interpréter.

Idée de l'interprétation. Avec Olivier Laurent, nous avons mis au point une interprétation des processus dans les réseaux différentiels dans laquelle on associe à chaque nom libre d'un processus deux ports libres du réseau qui l'interprète : un port d'émission et un port de réception. Dans une mise en parallèle, les réseaux sont alors connectés entre eux au moyen de *zones de communication*.

Nous avons adopté, pour ces réseaux d'interaction différentiels, un système de types inspiré du lambda-calcul pur : ce système est basé sur un type *sortie* o, satisfaisant l'*équation récursive* $o = ?(o^\perp) \parr o$ ou, ce qui revient au même, en posant $\iota = o^\perp$, l'équation $\iota = !o \otimes \iota$. Dans les figures qui suivent, le type associé à un fil orienté sera toujours $?\iota$, sauf s'il est indiqué explicitement. Toutes les paires de fils associées aux noms dans notre interprétation des processus portent les types duaux $?\iota$ et $!o$.

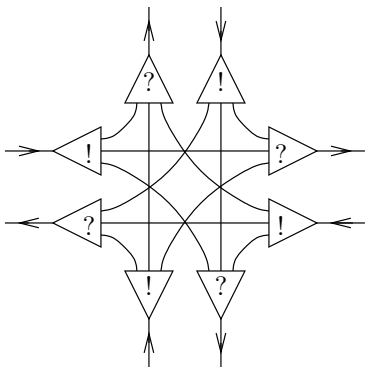
La zone de communication typique est la structure



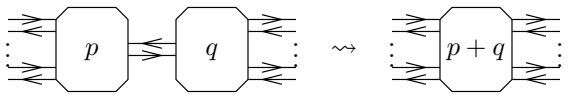
où les paires de ports à connecter à celles des réseaux à mettre en parallèle sont mises en évidence (on voit bien qu'elles portent des types duaux). Cette zone de communication est celle d'arité 3, on en construit une pour chaque arité. Celle d'arité 4 comporte 4 co-contractions et 4 contractions ayant chacune 3 ports auxiliaires [37] :

36. En particulier, des occurrences sujet de noms pourront être substituées dans cette étape. Cela signifie que les communications correspondantes auront lieu sur des canaux qui ne sont déterminés que par l'interaction en cours. Cette «mobilité» des processus est une des sources de la puissance expressive du pi-calcul.

37. En toute rigueur, ce ne sont pas des cellules mais des composées de cellules.



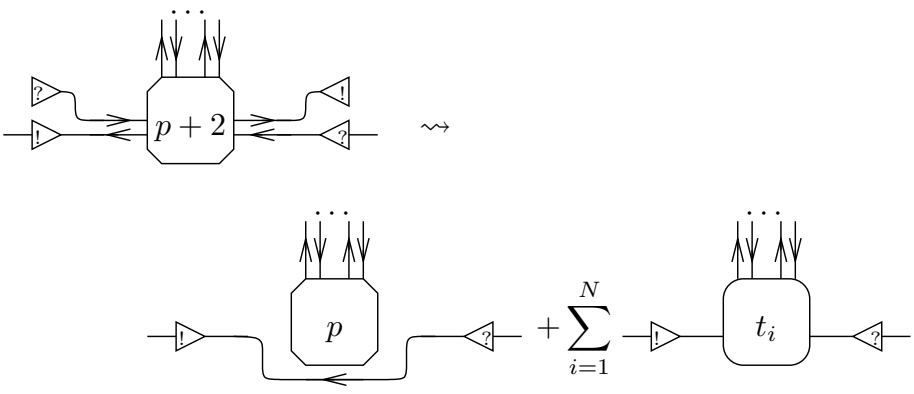
Ces zones de communication ont d'excellentes propriétés algébriques. Lorsqu'on connecte une zone à une autre, on obtient une nouvelle zone de communication en appliquant les règles de réduction structurelles : si on appelle *ordre* d'une zone de communication le nombre qui est égal à son arité moins 2, on a la loi suivante d'agglomération des zones communications :



où les zones de communication sont représentées par des rectangles biseautés, étiquetés par l'ordre de la zone en question. Cela signifie que l'on peut passer du membre gauche au membre droit, en appliquant uniquement les règles de bases de réductions des réseaux d'interaction différentiels que nous avons données.

Les préfixes du pi-calcul sont interprétés au moyen de la codéréliction (pour le récepteur) et de la déréliction (pour l'émetteur), ainsi qu'au moyen des cellules multiplicatives (tenseur et par), pour «mettre ensemble» (*multiplexer*) les paires de fils correspondant aux noms à émettre dans les préfixes d'émission, et les paires de fils correspondant aux noms liés dans les préfixes de réception. On utilise également les connecteurs multiplicatifs pour implémenter le fait que les préfixes ne peuvent communiquer que s'ils sont en «position extérieure» (c'est-à-dire, s'ils ne sont pas gardés par d'autres préfixes) : la partie multiplicative de la logique linéaire différentielle s'est révélée essentielle dans le codage des aspects *synchrones* du pi-calcul.

Voici ce qui se passe lorsqu'une déréliction (correspondant à un préfixe d'émission) et une codéréliction (préfixe de réception) se retrouvent en contact avec la même zone de communication :



Après un certain nombre d'étapes de réduction, on voit que, dans le premier terme de la somme, la déréliction et la codéréliction se retrouvent en contact direct (dans une étape successive du calcul, elles disparaîtront toutes deux, et leurs ports auxiliaires se retrouveront en contact direct : la communication entre les deux sous-processus correspondant sera établie), et il reste une zone de communication, d'arité inférieure de 2, sur laquelle le reste du réseau reste connecté, comme avant la réduction. Les autres N termes du résultat correspondent à des situations où, au lieu de communiquer entre eux, la déréliction et le codéréliction communiquent justement avec le reste du réseau. Cette interaction de base est l'idée fondamentale qui permet de simuler la réduction du pi-calcul dans les réseaux d'interaction différentiels.

Ce travail a donné lieu à [EL07] (CONCUR 2007), suivi par une version longue [EL10b].

Interprétation du calcul des solos. On s'est également intéressé à un autre calcul de processus, le *calcul des solos*, qui est totalement asynchrone contrairement au pi-calcul et a une notion de calcul plus «globale» [38] que ce dernier. On a développé une traduction très directe des processus solos dans les réseaux d'interaction différentiels, et on a analysé une traduction du pi-calcul dans les solos, variation sur une traduction de Laneve, Parrow et Victor [LPV01]. On a trouvé des conditions d'acyclicité sur les processus solos, qui

— sont satisfaites par les processus solos venant de la traduction de processus pi-calcul,
— sont préservées par réduction des processus solos
— et assurent une forme d'acyclicité des réseaux différentiels obtenus en les traduisant.

Ces résultats sont présentés dans [EL10a]. C'est en analysant la composition de ces deux traductions que nous avons eu l'idée de l'utilisation des multiplicatifs pour coder les aspects synchrones du pi-calcul (paragraphe ci-dessus).

Ces travaux sur les calculs de processus représentés de façon graphique m'ont mené à une collaboration avec Jiang Ying qui continue aujourd'hui.

4.6 Un CCS pour les arbres (depuis 2009)

A l'occasion de mon séjour Pékin en hiver 2009-2010, j'ai commencé à travailler avec Jiang Ying (Académie des Sciences) sur un nouveau calcul de processus similaire au CCS de Milner mais qui, au lieu d'être basé sur les automates ordinaires (reconnaissant des mots), est basé sur les automates d'arbres.

Une bonne façon de comprendre CCS est en effet de le voir comme une *clôture interactive* des automates finis et nous proposons une clôture similaire pour les automates d'arbres [CDG+07]. Notre formalisme CCTS se fonde sur des idées antérieures de Jiang et de ses élèves (représentation syntaxique des automates d'arbres) et sur les intuitions acquises lors de mon travail avec Laurent sur la représentation des calculs de processus dans les réseaux d'interaction (4.5.4).

Par rapport au CCS ordinaire de Milner, la nouveauté réside dans le fait que la composition parallèle n'a plus pour effet de mettre tous les processus en contact avec tous les autres : elle établit des liens entre les processus suivant un graphe non orienté qui évolue au cours de la réduction.

38. Dans les récepteurs des solos, les noms sujets *ne sont pas liés* : il y a symétrie complète entre émetteurs et récepteurs. Les émetteurs et les récepteurs sont des *solos*, appelés ainsi parce qu'ils ne préfixent pas de sous-processus, contrairement à ce qui se passe dans le pi-calcul. Lorsqu'un émetteur communique avec un récepteur, il en résulte une *égalisation* (on dit aussi *fusion*) des noms sujets, qui est globale à l'ensemble du processus, et permettra d'autres communications de se faire, des sujets de solos émetteurs et récepteurs, jusqu'ici différents, pouvant devenir égaux suite à cette fusion de noms.

Nous avons défini une sémantique opérationnelle de réécriture pour ce calcul, qui induit une équivalence observationnelle sur les processus (*barbed weak congruence*, bwc). Nous avons aussi défini une bisimilarité faible, dont nous avons prouvé qu'elle entraîne la bwc, et nous avons montré que ce formalisme étend à la fois CCS et les automates d'arbre, et que notre bisimilarité généralise la bisimilarité faible ordinaire de CCS.

Plus récemment (automne 2014), nous avons utilisé ce formalisme, en l'appliquant à des processus particuliers de structure arborescente, pour définir une nouvelle notion d'inclusion d'arbres qui semble avoir un lien avec les *modèles de mémoire faibles*. Nous espérons que ce travail mènera à des applications de CCTS à ce domaine de recherche appliqué.

Ce travail fait l'objet de l'article [EJ13] soumis à un journal pour publication.

4.7 Sémantique relationnelle (depuis 2006)

Cette sémantique est sous-jacente à la plupart des modèles de la logique linéaire que j'ai considérés jusqu'ici. Mon travail sur la logique linéaire différentielle en a renforcé l'importance à mes yeux. En effet, tout point de l'interprétation relationnelle d'une formule de la logique linéaire A peut être vu comme une preuve, dans un système de logique linéaire différentielle un peu étendu, de A et aussi de $A^\perp$. Il peut sembler curieux qu'on ait à la fois des preuves d'une formule et de sa négation ; cela vient du fait que les «preuves» que nous considérons ne sont pas totales, on les appelle parfois des «parapreuves» car ce ne sont donc pas des vraies preuves au sens de la logique, mais elles en ont toutes les propriétés du point de vue de l'élimination des coupures. La question de savoir si un point a de l'interprétation de A appartient à l'interprétation d'une preuve π (lambda-terme ordinaire, différentiel, réseau…) close de type A se ramène alors à l'élimination de la coupure entre π et la preuve de $A^\perp$ associée à a dans la logique linéaire différentielle : soit cette élimination des coupures se termine avec le réseau vide, et alors a est dans l'interprétation de π, soit elle se termine avec 0, et alors ce n'est pas le cas. Nous avons récemment mis ce fait à profit dans [BCEM11], voir 4.7.7.

Il n'y a donc pas de frontière entre la logique linéaire différentielle et sa sémantique relationnelle, d'où l'intérêt d'étudier celle-là pour mieux comprendre celle-ci.

4.7.1 Quelques mots sur la sémantique relationnelle.

La catégorie **Rel** a pour objets les ensembles et, si E et F sont des ensembles, $\mathbf{Rel}(E,F) = \mathcal{P}(E \times F)$. La composition est la composition ordinaire des relations, l'identité est la relation diagonale.

C'est une catégorie monoïdale, avec comme produit tensoriel $E \otimes F = E \times F$ (qui n'est pas un produit cartésien, dans cette catégorie), le neutre du tenseur est 1, le singleton. Elle est monoïdale fermée, avec $E \multimap F = E \times F$ comme espace des morphismes de E vers F. La catégorie **Rel** est $*$-autonome (au sens de [Bar79]), avec $\perp = 1$ comme objet dualisant, si bien que $E^\perp$ (le dual de E, qui est $E \multimap \perp$) est (trivialement isomorphe dans **Rel** à) E. Elle a tous les produits cartésiens : le produit d'une famille d'ensembles $(E_i)_{i\in I}$ est leur union disjointe $\bigcup_{i\in I}(\{i\} \times E_i)$, et c'est aussi leur somme (au sens catégorique).

Enfin, on a un foncteur $! : \mathbf{Rel} \to \mathbf{Rel}$ qui envoie un ensemble E sur l'ensemble $!E$ des multiensembles finis d'éléments de E, et un morphisme $t \in \mathbf{Rel}(E,F)$ sur $!t \in \mathbf{Rel}(!E,!F)$ qui est l'ensemble des couples de multiensembles $([a_1,\dots,a_n],[b_1,\dots,b_n])$ où $n \in \mathbf{N}$ et $(a_i,b_i) \in t$ pour tout i. Ce foncteur a une structure de comonade, ce qui fait de **Rel** un modèle de la logique linéaire. La catégorie de Kleisli de cette comonade, dont les objets sont les ensembles, et où un morphisme

de E vers F est un morphisme (c'est-à-dire ici, une relation) de $!E$ vers F, est cartésienne fermée. Les identités et la composition de cette catégorie sont définies en utilisant la structure de comonade du foncteur «!».

La sémantique relationnelle est souvent considérée comme dégénérée, car elle interprète par le même ensemble une formule et sa négation linéaire. Cependant, elle est absolument non triviale, puisque les preuves n'y sont pas du tout identifiées. Un résultat récent de De Carvalho et Tortora de Falco montre même que cette sémantique identifie le moins possible les preuves (théorème d'injectivité de la sémantique relationnelle) : essentiellement, deux réseaux normaux distincts de la logique linéaire ont des interprétations distinctes dans la sémantique relationnelle.

Une remarque curieuse : dans les espaces cohérents et dans les hypercohérences, on peut définir $!X$ comme l'ensemble des cliques finies de X muni d'une certaine relation de cohérence. On serait tenté de copier cela dans la sémantique relationnelle, et de définir $!E$ comme l'ensemble des parties finies de E. Mais cette interprétation ne fonctionne pas (on arrive à définir un foncteur, mais on ne parvient pas à lui donner la structure de comonade voulue). En essayant de corriger le problème, on est amené à munir les ensembles d'une relation de préordre et à considérer les ensembles clos vers le bas pour ce préordre, ce qui nous ramène à... la sémantique de Scott, voir 4.7.3.

4.7.2 Un modèle relationnel du lambda-calcul pur

Il s'agit d'un travail commun avec Bucciarelli et Manzonetto.

Nous avons découvert un modèle du lambda-calcul pur dans la catégorie de Kleisli du modèle de la logique linéaire des ensembles et des relations, qui est une catégorie cartésienne fermée.

Ce modèle du lambda-calcul pur est très simple à décrire : c'est le plus petit ensemble D qui est égal à l'ensemble des suites $(m_1, m_2, \dots)$ de multi-ensembles finis d'éléments de D, tels que tous les m_i soient le multi-ensemble vide, sauf peut-être un nombre fini d'entre eux. Autrement dit, les éléments de D sont des arbres finis qui alternent des couches où les branches sont indexées par les entiers, et des couches «commutatives», où les sous-arbres forment des multi-ensembles finis. Dans ce modèle, un lambda-terme clos est interprété comme un sous-ensemble de D.

Présentation algébrique du modèle. Dans un premier temps, nous avons montré comment ce modèle, bien que construit dans une catégorie qui n'a pas assez de points, peut être décrit au moyen des outils développés *a priori* pour des catégories ayant assez de points. Ce travail fait l'objet de [BEM07], présenté à la conférence internationale Computer Science Logic 2007 (CSL 2007). Dans le même papier, on a montré comment ce modèle permet également d'interpréter le lambda-mu-calcul pur (une version classique — au sens de la logique classique — du lambda-calcul).

Nous avons ensuite montré que la théorie équationnelle de ce modèle (c'est-à-dire, la théorie équationnelle qu'il induit sur les lambda-termes purs) est la théorie $\mathcal{H}^*$, qui correspond à l'égalité des arbres de Nakajima associés. L'arbre de Nakajima d'un lambda-terme est assez difficile à décrire, mais l'intuition est simple : c'est une forme normale infinie (au sens de la beta-réduction, qui transforme $(\lambda x M)N$ en $M[N/x]$), infiniment eta-expansée, l'eta-expansion étant une règle qui transforme un M en $\lambda x(M)x$, si x n'est pas libre dans M. Autrement dit, cette règle exprime que tout lambda-terme M est une fonction (celle qui à x associe M appliqué à x, bien sûr). De ce point de vue équationnel, ce modèle D est tout à fait similaire au D_∞ de Dana Scott, bien que beaucoup plus simple à décrire.

Adéquation pour un calcul avec composition parallèle. Ce modèle permet d'interpréter facilement des versions non déterministes du lambda-calcul pur : plus précisément, on dispose de deux opérations dans $\mathcal{P}(D)$, à savoir une somme (la réunion) et un produit, qui vient du fait qu'à toute paire (d, d') d'éléments de D, on peut associer l'élément de D dont le i-ème élément est la somme des multi-ensembles d_i et d'_i. En dernière analyse, cette opération est liée à la règle *MIX* de la logique linéaire, et un lambda-mu-calcul étendu par cette règle a déjà été considéré par Danos et Krivine dans [DK00], qui ont montré qu'elle permet de réaliser la formule $(A \to B) \vee (B \to A)$. Cette formule est vraie en logique classique mais pas en logique intuitionniste, et son contenu algorithmique est lié au mécanisme informatique des *coroutines*.

Nous avons donc considéré un lambda-calcul non déterministe et parallèle, qui est le lambda-calcul pur ordinaire étendu avec deux opérations binaires sur les termes, dont la seconde distribue sur la première, et tel que l'application distribue sur les deux opérations, uniquement lorsqu'elles sont appliquées du côté *fonction* de l'application. Ce lambda-calcul non déterministe et parallèle s'interprète naturellement dans ce modèle. On a étendu à ce calcul certains résultats classiques du lambda-calcul pur, notamment le théorème de forme normale de tête (qui énonce que si un lambda-terme a une forme normale de tête, cette dernière peut être obtenue par une stratégie de réduction particulière, appelée réduction de tête), qui se démontre élégamment par une méthode de réalisabilité dans laquelle le rôle des types est joué par les éléments de D (comme dans les *types avec intersection* largement étudiés par l'école turinoise de lambda-calcul). Ces résultats font l'objet de [BEM09].

4.7.3 Sémantique de Scott de la logique linéaire

On peut développer la sémantique de Scott (voir 4.1.1) dans la catégorie des treillis complets (qui sont des ordres partiels complets particuliers), et en particulier dans la catégorie des treillis complets premier-ω-algébriques (qui ont un nombre au plus dénombrable d'éléments premiers et où tout élément est le sup de ses minorants premiers). On sait aussi que tout tel treillis peut être représenté au moyen d'un préordre partiel au plus dénombrable : un élément du treillis est un sous-ensemble du préordre, clos vers le bas pour la relation de préordre (on pourrait remplacer les préordres par des ordres, mais cela compliquerait le traitement des exponentielles). Ces préordres jouent le rôle de *trames* vis-à-vis des treillis considérés.

Huth [Hut93] et, indépendamment, Winskel [Win04] (mais le premier à avoir fait informellement cette remarque est probablement Krivine, dès l'introduction de la logique linéaire par Girard, et il faudrait aussi citer l'observation faite par Barr [Bar79] que les treillis complets forment une catégorie $*$-autonome) ont montré que la catégorie de ces treillis, avec les fonctions *linéaires* (commutant à tous les sups) comme morphismes forme un modèle de la logique linéaire et que les constructions logiques s'interprètent de façon plus simple si on les considère sur les préordres partiels plutôt que directement sur les treillis.

Concrètement, on définit une catégorie dont les objets sont des couples $X = (|X|, \leq_X)$ où $|X|$ est un ensemble au plus dénombrable et $\leq_X$ est une relation de préordre sur $|X|$. On définit $\mathcal{I}(X)$ comme l'ensemble des parties de $|X|$ qui sont closes vers le bas pour $\leq_X$: c'est un treillis complet premier-algébrique. Un morphisme de X vers Y dans cette catégorie est une fonction linéaire de $\mathcal{I}(X)$ vers $\mathcal{I}(Y)$. Cette catégorie forme un modèle de la logique linéaire : la négation linéaire $X^\perp$ de X est $(|X|, \geq_X)$ (on renverse le préordre), $X \otimes Y$ est $|X| \times |Y|$ muni du préordre produit, $X \,\&\, Y$ est l'union disjointe de $|X|$ et $|Y|$ muni de l'union des préordres. Pour l'exponentielle $!X$, on peut prendre l'ensemble des multiensembles finis d'éléments de $|X|$, avec le préordre défini par : $m \leq_{!X} m'$

si tout élément de m a un majorant dans m' (sans tenir compte des multiplicités). En prenant des ensembles finis au lieu de multiensembles finis (et la même définition pour le préordre), on aurait obtenu le même treillis $\mathcal{I}(!X)$, et c'est le choix qu'ont fait les auteurs mentionnés ci-dessus. Dans la catégorie de Kleisli associée à cette exponentielle, les morphismes de X vers Y sont exactement les fonctions Scott-continues : on retrouve la sémantique de Scott usuelle évoquée dans l'introduction, voir 4.1.1.

Grâce à cette définition de $!X$, cette version de la sémantique de Scott de la logique linéaire a la même propriété que celle des espaces cohérents non uniformes, des espaces de Köthe, des espaces de finitude etc. : la trame d'un objet interprétant une formule de la logique linéaire coïncide avec l'interprétation de cette formule dans le modèle relationnel présenté ci-dessus. Le modèle de Scott apparaît donc comme un enrichissement du modèle relationnel dans lequel tout objet (ensemble) est muni d'une relation de préordre, et il devient possible de comparer les deux modèles. J'ai souhaité comprendre le lien entre ces deux modèles car le modèle relationnel sert de fondement à la sémantique stable, fortement stable etc., et la sémantique de Scott est un outil fondamental dans l'étude des langages de programmation. Mon but était aussi de comprendre s'il est possible d'interpréter le lambda-calcul avec ressources dans la sémantique de Scott de façon compatible avec la sémantique relationnelle, question à laquelle j'ai obtenu une réponse négative.

Cette légère modification dans la présentation de la sémantique de Scott, la plaçant elle aussi dans le cadre relationnel standard, rendait possible l'étude du lien entre sémantique de Scott et sémantique relationnelle.

4.7.4 Collapse extensionnel de la sémantique relationnelle

La catégorie de Kleisli du modèle relationnel n'est pas extensionnelle (au sens de la note 13). J'ai montré que le modèle de Scott est le collapse extensionnel (voir encore la note 13) du modèle relationnel. Pour cela, il faut munir les objets du modèle relationnel d'une relation d'équivalence partielle, ou plus précisément, construire un modèle dont les objets sont des paires $E = (|E|, \sim_E)$ où $|E|$ est un ensemble au plus dénombrable et $\sim_E$ est une relation d'équivalence partielle sur $\mathcal{P}(|E|)$ et montrer qu'en un certain sens, le quotient $\mathcal{P}(|E|)/\sim_E$ est isomorphe à $\mathcal{I}(X)$, quand E et X sont l'interprétation de la même formule (X étant l'interprétation dans le modèle de Scott). J'ai développé le formalisme qui permet de montrer ce résultat, dans le cadre de la logique linéaire (et pas seulement le cadre intuitionniste qui aurait suffi pour établir le résultat de collapse pour la catégorie cartésienne fermée des ensembles et des relations, cf. le point méthodologique expliqué en 4.1, page 5).

L'idée est de considérer des objets «mixtes» $X = (|X|, \leq_X, \mathsf{D}(X))$, où $(|X|, \leq_X)$ est un préordre (objet de la sémantique de Scott de la logique linéaire) et $\mathsf{D}(X) \subseteq \mathcal{P}(|X|)$ correspond, intuitivement, aux parties de $|X|$ qui sont équivalentes à elles-mêmes au sens de la relation d'équivalence partielle du collapse (dans le modèle relationnel donc) ; on exploite le fait que la trame du préordre interprétant une formule dans la sémantique de Scott coïncide avec son interprétation relationnelle. Encore une fois, $\mathsf{D}(X)$ doit être saturé par rapport à une relation de dualité : on dit ici que $u, u' \subseteq |X|$ sont en dualité si, dès que u' contient un minorant d'un élément de u au sens de $\leq_X$, il contient aussi un élément de u (noter que cette définition est symétrique en u et u', à condition de renverser le préordre $\leq_X$). C'est une forme de dualité nouvelle, qui, je crois, n'avait jamais été rencontrée en sémantique dénotationnelle.

Si, étant donné $\mathcal{D} \subseteq \mathcal{P}(|X|)$, on note comme d'habitude $\mathcal{D}^\perp$ l'ensemble de tous les $u' \subseteq |X|$ qui sont en dualité avec tout élément de $\mathcal{D}$, on demande que $\mathsf{D}(X) = \mathsf{D}(X)^{\perp\perp}$. On peut alors définir

la relation d'équivalence du collapse en disant que deux sous-ensembles de $|X|$ sont équivalents s'ils appartiennent tous deux à $\mathsf{D}(X)$ et ont même clôture vers le bas dans le préordre $\leq_X$.

En montrant que ces objets mixtes forment un modèle de la logique linéaire, on obtient le théorème voulu de collapse extensionnel. Ce théorème s'étend aux objets réflexifs, et on montre aussi que le modèle D de 4.7.2 a un collapse extensionnel dans le modèle de Scott, dont on peut montrer que c'est D_∞ de Scott (cela a été prouvé lors d'un stage de Master 2 LMFI en 2010).

Ces résultats sont publiés dans [Ehr12b].

Application du collapse à la normalisation. Les systèmes de types avec intersection peuvent être vus comme une façon syntaxique de présenter des modèles dénotationnels du lambda-calcul (et de ses extensions : lambda-mu calcul, PCF etc). On distingue les types avec intersection idempotents, qui correspondent à des sémantiques à la Scott et les types non-idempotents, qui correspondent à des sémantiques relationnelles. Des théorèmes de normalisation sont associés à ces système de types. Dans le cas relationnel, ces théorèmes se prouvent de façon combinatoire et se ramènent, à travers le développement de Taylor 4.5.2, à la normalisation forte du lambda-calcul avec ressources. Dans le cas idempotent, cet argument combinatoire n'est pas disponible et la preuve utilise ordinairement des arguments de réalisabilité. On a montré comment utiliser le théorème de collapse extensionnel ci-dessus pour ramener la normalisation idempotente à la normalisation non-idempotente : la réalisabilité est ainsi toute entière contenue dans construction sémantique de la catégorie mixte de collapse. Le papier [Ehr12a] présente cette nouvelle approche dans le cas du lambda-calcul par valeurs (mais ce n'est qu'un exemple).

4.7.5 Exponentielles infinitaires (2010)

Dans le cadre de l'étude générale des modèles du lambda-calcul différentiel entrepris avec Carraro et Salibra (voir 4.8.2), nous avons réalisé que, dans les catégories cartésiennes fermées différentielles où la formule de Taylor est vraie (en pratique, tous les modèles du lambda-calcul différentiel considérés jusqu'ici), un modèle du lambda-calcul pur est forcément «sensible» (terminologie anglaise qui n'a pas de traduction française officielle à ma connaissance), ce qui signifie que tous les termes non résolubles y sont identifiés. Or on sait depuis longtemps construire des modèles non sensibles du lambda-calcul pur, dans la sémantique de Scott, ou dans la sémantique stable, ou fortement stable...

Ces constructions semblent utiliser fortement le fait que, dans ces modèles, on peut décrire l'exponentielle au moyen d'*ensembles* finis, et non de multiensembles finis. Plus précisément, la construction de tels modèles utilise le fait que la réunion est une opération idempotente sur les ensembles, ce qui n'est pas le cas de la somme des multiensembles. Or on sait que les exponentielles ensemblistes ne permettent pas de modéliser le lambda-calcul différentiel, et, d'autre part, les modèles à base d'exponentielle multiensembliste semblent condamnés à satisfaire la formule de Taylor.

Nous avons trouvé une issue à cette contradiction apparente en généralisant la notion de multiensemble fini. Un multiensemble fini d'éléments d'un ensemble I est une fonction $I \to \mathbf{N}$ qui envoie presque tout élément de I sur 0. Nous avons analysé les propriétés de $\mathbf{N}$ qui sont nécessaires pour construire un modèle de la logique linéaire différentielle et sommes ainsi arrivés à un ensemble d'axiomes définissant une notion de *semi-anneau de multiplicité* : c'est un semi-anneau commutatif unitaire $\mathbb{M}$ tel que

1. $n_1 + n_2 = 0 \Rightarrow n_1 = n_2 = 0$
2. $n_1 + n_2 = 1 \Rightarrow n_1 = 0$ ou $n_2 = 0$
3. $n_1 + n_2 = m_1 + m_2 \Rightarrow \exists q_{11}, q_{12}, q_{21}, q_{22}\ n_i = q_{i1} + q_{i2}$ et $m_j = q_{1j} + q_{2j}$
4. $n_1 + n_2 = pm \Rightarrow \exists p_1, p_2, m_{11}, m_{12}, m_{21}, m_{22}\ p = p_1 + p_2,\ m = m_{11} + m_{21} = m_{12} + m_{22}$ et $n_i = p_1 m_{i1} + p_2 m_{i2}$ pour $i = 1, 2$.

Le premier axiome exprime que les éléments de $\mathbb{M}$ sont «positifs», le deuxième, que 1 est indécomposable, le troisième peut être vu comme stipulant l'existence d'un raffinement commun lorsqu'un même ensemble est partitionné de 2 façons distinctes. Le dernier, plus complexe mais absolument essentiel, n'est pas sans rapport avec la division euclidienne.

L'ensemble $\mathbf{N}$ des entiers naturels satisfait évidemment ces axiomes (avec les opérations habituelles), mais on a exhibé des semi-anneaux de multiplicité contenant des éléments ω infinis au sens où $\omega + 1 = \omega$. L'exemple le plus simple est l'ensemble des entiers naturels complété $\mathbf{N} \cup \{\omega\}$, avec $\omega + n = \omega$, $\omega 0 = 0$ et $\omega n = \omega$ si $n \neq 0$. On a trouvé aussi d'autres exemples plus subtils (sans élément idempotent).

On a montré que, si on définit une exponentielle sur les ensembles en prenant les multiensembles finis à valeur dans $\mathbb{M}$ (au lieu de $\mathbf{N}$), on obtient encore un modèle de la logique linéaire, et qu'il est possible, dans la catégorie de Kleisli associée, qui est une CCC différentielle *où la formule de Taylor n'est pas vraie*, de construire un modèle non sensible du lambda-calcul pur, dès lors que $\mathbb{M}$ contient un élément ω tel que $\omega + 1 = \omega$. Ces résultats font l'objet de [ECS10].

La notion de semi-anneau de multiplicité semble avoir son propre intérêt. Les preuves sur ces objets sont assez complexes (car l'axiomatique n'est pas purement équationnelle), et nous avons utilisé une méthode graphique pour mener les calculs, basée sur les réseaux d'interaction, et qui mériterait sûrement d'être davantage explorée. Enfin, il serait important de comprendre si ces axiomes sont vraiment nécessaires, ce qui revient à comprendre la structure générale d'une exponentielle dans le modèle relationnel de la logique linéaire.

4.7.6 Classification pour les exponentielles du modèle relationnel (2010)

Ces résultats montrent qu'il y a plus de liberté que ce que la plupart des gens imaginaient dans la définition des exponentielles (satisfaisant tous les axiomes catégoriques requis) dans le cadre du modèle relationnel. J'ai commencé à étudier la question de classifier toutes les exponentielles dans le cadre du modèle relationnel, l'idée étant d'associer à chaque exponentielle une structure algébrique (et peut-être bi-algébrique) généralisant les semi-anneaux de multiplicité. Ce travail, commencé en 2010 dans le cadre de la thèse de Sarah Bulteau a été interrompu pas l'abandon de celle-ci.

Ce thème a été repris par Flavien Breuvart dans sa thèse de doctorat (soutenance à l'automne 2015), avec un point de vue plus opérationnel très innovant.

4.7.7 Complète adéquation pour un calcul avec ressources différentiel (2010)

Avec Bucciarelli, Carraro et Manzonetto, nous nous sommes intéressés à la sémantique du lambda-calcul différentiel (ou avec ressources) dans le modèle relationnel D évoqué en 4.7.2. En étendant la syntaxe de ce calcul avec deux constructions correspondant aux cellules «tenseur vide» et au «par vide» que j'avais introduites dans les réseaux d'interaction différentiels avec Laurent [EL10b] pour représenter les calculs de processus, et avec une «composition parallèle» correspondant à la règle MIX de la logique linéaire. À l'automne 2010, nous avons montré que tous les éléments de D

sont définissables dans ce langage et nous en avons déduit un théorème d'adéquation complète qui exprime que la relation d'ordre que le modèle induit sur la syntaxe coïncide avec l'ordre observationnel, défini syntaxiquement à partir de la réduction. Ce travail a fait l'objet d'un article accepté à la conférence internationale CSL 2011 [BCEM11], et fait l'objet d'un article plus détaillé [BCEM12]. Il sert également de point de départ au travail de thèse de Flavien Breuvart, sous la direction de Bucciarelli et Pagani.

4.8 Axiomatique des modèles des lambda-calculs différentiels (depuis 2009)

Il s'agit ici de comprendre les conditions générales que doit satisfaire une structure, catégorique ou algébrique, pour être un modèle du lambda-calcul différentiel (ou du lambda-calcul avec ressources).

4.8.1 Catégorie cartésiennes fermées différentielles (2010)

Dans le but de mieux comprendre l'axiomatique catégorique des catégories cartésiennes fermées dans lesquelles l'opération de différentiation est possible (comme par exemple les catégories de Kleisli du modèle relationnel, du modèle des espaces de Köthe, des espaces convénients, des espaces de finitude...), nous avons prolongé le travail de Blute, Cockett et Seely sur les catégories cartésiennes différentielles, aboutissant à une notion de catégorie cartésienne fermée différentielle qui fournit probablement l'axiomatique la plus générale de la situation considérée. Ce travail fait l'objet de [BEM10].

4.8.2 Algèbres combinatoires avec ressources (2010)

Lors de mon invitation à Venise en février-mars 2010, nous avons travaillé avec Salibra et Carraro sur une généralisation des algèbres combinatoires et des algèbres de lambda-abstraction au lambda-calcul différentiel et au lambda-calcul avec ressources. Ces algèbres permettent d'éviter le formalisme catégorique. Par le passé, elles ont permis à Salibra d'obtenir des résultats spectaculaires sur le classification des modèles du lambda-calcul pur et sur l'incomplétude de certaines classes naturelles de modèles (continus, stables, fortement stables) en utilisant des techniques venant de l'algèbre universelle.

Dans le but de développer une approche purement algébrique des modèles du lambda-calcul différentiel, nous avons introduit une notion d'algèbre de lambda-abstraction différentielle dans [CES10], et montré son équivalence avec le formalisme catégorique général.

4.8.3 Calcul de primitives dans la logique linéaire différentielle (depuis 2011).

Dans [Ehr11], je présente une notion générale de modèle pour la logique linéaire différentielle sans promotion, qui consiste en une catégorie $*$-autonome, de produit tensoriel noté $\otimes$, enrichie au-dessus des $\mathbf{k}$-modules (où $\mathbf{k}$ est un semi-anneau dont les éléments sont les coefficients utilisés dans la syntaxe) munie d'une opération qui, à tout objet X, associe une $\otimes$-bialgèbre $!X$ munie d'un morphisme $\mathrm{d}^X : !X \to X$ et d'un morphisme $\overline{\mathrm{d}}^X : X \to !X$ qui satisfont des équations correspondant aux règles de réduction de la logique linéaire différentielle. Cette opération $X \mapsto !X$ n'est pas supposée fonctorielle, même si une forme faible de fonctorialité peut être déduite de cette structure.

En utilisant la comultiplication (contraction) et la multiplication (co-contraction) de ces bigèbres et bien sûr d^X et $\overline{\mathrm{d}}^X$, il est possible de définir des morphismes $\partial^X : !X \to !X \otimes X$ et $\overline{\partial}^X : !X \otimes X \to !X$. Étant donné $f : !X \to Y$, qui doit être considéré comme un morphisme non linéaire de X vers Y, par exemple, une fonction entière, analytique etc, la dérivée de f est alors simplement donnée par composition : $f' = f \circ \overline{\partial}^X : !X \otimes X \to Y$. De même, si $g : !X \otimes X \to Y$, on peut définir la dérivée $g' : !X \otimes X \otimes X \to Y$. On obtient ainsi la dérivée seconde $f'' : !X \otimes X \otimes X \to Y$ de f, qui est symétrique par rapport à ses deux derniers paramètres, autrement dit $f'' \circ (\mathrm{Id}_{!X} \otimes \sigma) = f''$ où $\sigma : X \otimes X \to X \otimes X$ est l'isomorphisme canoniquement associé à la symétrie du produit tensoriel.

Étant donné $g : !X \otimes X \to Y$ tel que $g' : !X \otimes X \otimes X \to Y$ vérifie la même condition de symétrie $g' \circ (\mathrm{Id}_{!X} \otimes \sigma) = g'$, il est très naturel de se demander dans ce cadre général s'il existe forcément un $f : !X \to Y$ tel que $g = f'$, autrement dit, si g admet une primitive. J'ai montré que c'était le cas à partir du moment où le morphisme $\mathrm{Id}_{!X} + (\overline{\partial}^X \circ \partial^X) : !X \to !X$ est un isomorphisme : sous cette hypothèse (vérifiée dans tous les modèles connus), on peut toujours construire f à partir de g en transposant la preuve classique du *Lemme de Poincaré*. Ce résultat est présenté dans [Ehr11] où je montre aussi comment faire la même chose dans le cadre, syntaxique cette fois, du calcul avec ressources sans promotion.

Je vois ce travail comme un premier pas dans la direction de l'étude d'équations différentielles dans le cadre de la logique linéaire différentielle et du contenu opérationnel de leurs solutions. J'ai commencé en septembre 2014 la direction d'une thèse sur ce sujet (celle de Marie Kerjean).

4.8.4 Syntaxe pour les réseaux (2013-2014)

Au cours de la rédaction de [Ehr11], j'ai été amené à introduire une syntaxe pour dénoter les réseaux de preuves, considérant que les dessins et les graphes auxquels nous sommes habitués depuis l'article originel [Gir87] ne sont pas complètement satisfaisants. Cette syntaxe est dérivée de celles introduites par différents auteurs, Mackie et Fernandez notamment [FM99]. Est apparue alors naturellement la nécessité de disposer d'un critère de correction qui soit adapté à ce type de présentation. J'ai trouvé un tel critère en août 2013, qui a fait l'objet d'un papier accepté à LICS-CSL 2014 [Ehr14], et j'ai découvert un mois avant la conférence (Vienne, juillet 2014) que le dit critère avait déjà été publié par Rétoré en 2003 [Ret03]. Je vois plusieurs aspects positifs à cette expérience plutôt désagréable :

- Elle m'a permis d'attirer l'attention de la communauté sur un critère de correction qui avait été oublié et qui est très intéressant et très différent des autres.
- Elle m'a permis de trouver une autre façon de prouver le critère de Danos-Regnier, en le ramenant à une propriété purement graphique que j'ai redécouverte (après Rétoré).
- Elle est à l'origine du modèle des ensembles pondérés de la logique linéaire que j'ai trouvé en juillet 2014, juste après la conférence en question, et que j'ai appliqué à une version classique de PCF, voir 4.10.1.

4.9 Espaces cohérents et PCF probabilistes (depuis 2008)

Une de mes motivations pour le travail sur les espaces de Köthe avait été l'article de Danos et Harmer sur les jeux probabilistes [DH00]. J'y suis revenu lors d'un travail récent avec Danos, au cours duquel nous avons repris la notion d'espace cohérent probabiliste (qui est proche de celle d'espace de Köthe 4.4.1, et que Girard esquisse dans [Gir04]). La définition est la même, à ceci près qu'on ne considère que des familles de réels positifs, et que la dualité «$\sum_{i\in I} |u_i u'_i|$ fini» est

remplacée par «$\sum_{i\in I} u_i u'_i \leq 1$». Cette modification apparemment mineure change énormément de choses. On retrouve les intuitions de théorie des domaines sur l'interprétation des types : les espaces considérés peuvent être munis d'une relation d'ordre pour laquelle ils sont complets : ce sont des cpo ω-continus.

On obtient un modèle de la logique linéaire formellement similaire à celui des espaces de Köthe, avec une différence majeure : tout endomorphisme de la catégorie de Kleisli associée admet un plus petit point fixe. Cela permet d'interpréter dans ce modèle une version probabiliste du langage PCF de Milner et de montrer un théorème d'adéquation au moyen d'une technique de relation logique introduite par Plotkin. Dans ce langage, en plus des opérations habituelles, on dispose d'une primitive de «pile ou face» de type booléen, qui rend *vrai* ou *faux* avec probabilité $\frac{1}{2}$. L'interprétation des entiers est l'ensemble des sous-distributions de probabilités sur les entiers naturels (les fonctions $\nu : \mathbf{N} \to [0,1]$ telles que $\sum_{n\in\mathbf{N}} \nu(n) \leq 1$). L'interprétation d'un terme M clos de type entier de notre PCF probabiliste est donc une telle sous-distribution ν de probabilités, et le théorème d'adéquation dit que $\nu(n)$ est la probabilité que l'exécution de M se termine avec la valeur n.

On a ensuite suggéré un cadre plus général pour de telles interprétations dénotationnelles, utilisant des espaces de Banach ordonnés, mais il s'agit surtout d'une piste pour de futures recherches.

Une propriété intéressante de cette interprétation est que l'intuition probabiliste, qui est évidente dans la sémantique du type des entiers naturels, est perdue aux autres types. Une façon de la retrouver pourrait être de démontrer un théorème de collapse entre les jeux de Danos-Harmer et les espaces cohérents probabilistes.

On a aussi construit un modèle du lambda-calcul pur dans les espaces cohérents probabilistes. Ce travail sur les espaces cohérents probabilistes fait l'objet de [DE11].

4.9.1 Complète adéquation pour PCF probabiliste (depuis 2012)

Avec Pagani et Tasson (PPS), nous avons prouvé à l'automne 2012 que ce modèle est pleinement adéquat pour PCF probabiliste. Cela signifie que si deux termes clos M et M' de même type sont observationnellement équivalents (pour tout contexte $C[\_]$ de type entier, les termes clos $C[M]$ et $C[M']$ ont même probabilité de se réduire en n'importe quel entier), alors ils ont même interprétation dans le modèle ; la réciproque est une conséquence du théorème d'adéquation prouvé dans [DE11]. Cette preuve de pleine adéquation utilise fortement le fait que les morphismes des la catégorie de Kleisli de ce modèle (où les termes de PCF sont interprétés) sont des fonctions analytiques. La méthode utilisée s'inspire de celle que nous avons mise en oeuvre en 4.7.7, bien que les opérations différentielles caractéristiques des calculs avec ressources ne soient pas disponibles dans le cas probabiliste. Le papier correspondant [ETP14] a été accepté et présenté à POPL 2014. Nous avons aussi écrit une version longue [ETP15] de ce travail, avec davantage de détails et d'exemples, et pour une version de PCF plus réalistes d'un point du point de vue de la programmation.

En effet, la version de PCF considérée dans [ETP14] était complètement *appel par nom* (CBN) et la conditionnelle utilsée avait la forme traditionnelle $\texttt{if}\, M\, \texttt{then}\, P\, \texttt{else}\, Q$ avec la sémantique suivante : M est un terme de type entier ; si son évaluation rend 0 alors on évalue P, sinon on évalue Q. Or il se pourrait que Q ait besoin de la valeur de M pour s'évaluer, mais dans un cadre probabiliste il n'est pas certain qu'une nouvelle évaluation de M rende la même valeur entière que la précédente. Dans ces conditions il semble impossible d'écrire un algorithme probabiliste sérieux !

Dans [ETP15], nous avons donc modifié la conditionnelle en sorte que la valeur de M (ou plus exactement, son prédécesseur) soit passée à Q qui doit donc prendre un paramètre de type entier. Cela revient à dire que, sur le type des entiers uniquement, on autorise une évaluation en *appel*

par valeur (CBV). Du point de vue de la logique linéaire, c'est parfaitement justifié par le fait que l'objet $\mathbf{N}_0$ qui interprète les entiers a une structure canonique de !-coalgèbre ce qui signifie que les morphismes et donc les morphimses de !-coalgèbres à valeur dans $\mathbf{N}_0$ sont librement duplicables et effaçables (ces morphismes correspondent aux "valeurs").

En poussant plus loin cette idée de distinguer des types "positifs" (comme $\mathbf{N}_0$) pour lesquels l'effacement et le duplication sont possibles de types généraux, on retrouve je crois l'essence de l'idée de *call by push value* introduite par Paul Levy [Lev99] il y a quelques années. J'ai développé cette remarque dans un travail qui a suivi celui-ci, voir 4.11.

Le cas du lambda-calcul pur. Avec Pagani et Tasson, nous nous sommes intéressés à ce modèle du lambda-calcul pur (voir [DE11]) pour démontrer un analogue non typé du théorème d'adéquation ci-dessus. Ce modèle est un espace cohérent probabiliste dont la trame coïncide avec le modèle relationnel D présenté en 4.7.2 et ses éléments sont hiérarchisés suivant leur «profondeur». Nous avons montré qu'en sommant les valeurs de l'interprétation d'un terme sur les points de niveau 2 de cette hiérarchie, on obtient la probabilité que la réduction gauche du terme converge vers une forme normale de tête (pour une extension du lambda-calcul pur avec un opérateur naturel de choix probabiliste). Ce travail fait l'objet de l'article [EPT11], présenté à LICS en 2011.

4.10 Logique classique (depuis 2011)

C'est un travail qui commence par une collaboration avec Bucciarelli (PPS), Salibra et Carraro (université Ca' Foscari, Venise) initiée lors de mon séjour à Venise en février 2011, dans un souhait de mieux comprendre la réalisabilité classique de Krivine. A l'origine, notre but était plus précisément d'être capable de calculer la sémantique dénotationnelle de la constante `call/cc` qui joue un rôle crucial dans la réalisabilité classique : on sait depuis le papier fondateur de Griffin [Gri90] que cette opération standard du langage Scheme admet la Loi de Peirce (la tautologie classique $((A \to B) \to A) \to A$, qui n'est pas vraie dans la logique intuitionniste) comme type naturel. Ce travail fondateur a ouvert la voie à l'extension de la correspondance de Curry-Howard à la logique classique, avec des extensions du lambda-calcul comme le lambda-mu calcul de Parigot [Par92] et le formalisme utilisé par Krivine (lambda-calcul étendu par une constante `call/cc`) qui lui permet d'associer des réalisateurs aux axiomes et théorèmes de la théorie des ensembles : ces programmes (dans ce lambda-calcul étendu, auquel il faut aussi ajouter des constantes supplémentaires, dans le cas des axiomes) ont une interprétation souvent très intéressante en terme de procédures système. Mais l'approche de Krivine est aussi une généralisation du *forcing* (Cohen) pour développer de nouveaux modèles de la théorie des ensembles.

Une meilleure compréhension dénotationnelle des extensions classiques du lambda-mu calcul permet de mettre au point de nouveaux modèles de ZF en réalisabilité classique dans lesquels les réalisateurs sont des éléments d'un modèle du lambda-calcul classique, et non plus des termes. C'est ce qu'on fait Krivine et Streicher, séparément, peu après que j'eus montré à Krivine comment retrouver call/cc dans une version stable de D_∞ de Scott. Les modèles de ZF ainsi obtenus sont intermédiaires entre les modèles de forcing usuels et les modèles de réalisabilité syntaxique habituellement considérés par Krivine. Des questions naturelles se posent à leur sujet, comme de savoir s'ils introduisent de nouveaux ordinaux. Leur étude est l'un des sujet de la thèse d'Hadrien Batmalle que j'ai commencé à co-diriger avec Jean-Louis Krivine en septembre 2014.

Pour revenir à notre travail avec Bucciarelli, Carraro et Salibra, nous avons mis au point en 2011 un *calcul de piles* qui simplifie le calcul de Parigot en ne gardant que la μ-abstraction et en

y ramenant la λ-abstraction, ce qui est rendu possible par la réification des piles (suites finies de termes) qui deviennent des objets de 1$^{\text{ère}}$ classe. Ce nouveau formalisme a permis mener à bien le calcul mentionné ci-dessus de l'interprétation de `call/cc` dans D_∞, car il a un lien très simple et direct avec la logique linéaire polarisée. Ce travail a été présenté au workshop LSFA12 et a été accepté pour publication dans les proceedings de ce workshop.

4.10.1 PCF classique (2014)

En 2014, j'ai continué d'explorer cette direction de recherche en m'intéressant à l'extension classique du langage PCF, en lien avec la co-direction de la thèse de Batmalle avec Jean-Louis Krivine. Cette extension avait déjà été considérée par Herbelin [Her97], et plus récemment par Blot et Riba [BR13], avec, dans les deux cas, un point de vue sémantique limité aux jeux.

Pour développer une sémantique catégorique plus générale, dans l'esprit de la logique linéaire polarisée et des espaces de corrélation [39] [Gir91] — cela signifie qu'on part d'un modèle catégorique de la logique linéaire $\mathcal{L}$ et qu'on interprète les termes dans la "sous-catégorie" de la catégorie de Kleisli $\mathcal{L}_!$ dont les objets sont les !-coalgèbres —, je me suis rendu compte que le point clef est de comprendre quelle structure de !-coalgèbres pouvait être associée à l'objet des entiers naturels, qui doit interpréter le type de base du langage. Si $\mathbf{N}_0$ est l'objet interprétant le type des entiers dans $\mathcal{L}_!$ (pour une interprétation de PCF ordinaire), un choix canonique est d'interpréter les entiers de PCF classique par $!({\mathbf{N}_0}^\perp)$ autrement dit, de choisir la coalgèbre libre engendrée par les entiers de base (ou plutôt, le dual des entiers ; cette dualité vient du fait qu'on a choisi de travailler avec des !-coalgèbre ce qui semble plus standard d'un point de vue catégorique). Cela revient à effectuer une CPS-traduction $M \mapsto M^*$ de PCF classique dans PCF ordinaire et d'interpréter un terme classique M par l'interprétation de M^* dans le modèle de PCF ordinaire. La question est donc de savoir s'il est possible de trouver des coalgèbres non libre (et, intuitivement, beaucoup plus "petites") que $!({\mathbf{N}_0}^\perp)$ pour interpréter les entiers, afin que le modèle de PCF classique ressemble le plus possible à un modèle de PCF ordinaire. La contrainte semble ici venir principalement de la nécessité d'interpréter la conditionnelle par un morphimse de coalgèbres.

Dans le cas où les entiers sont interprétés par $!({\mathbf{N}_0}^\perp)$, un terme clos M de type entier sera interprété dans le modèle **Rel** par un ensemble de multi-ensembles finis d'entiers (un point de $?\mathbf{N}_0$) :

- La présence de plusieurs multiensembles correspond au fait que **Rel** est un modèle du non-déterminisme [40]. Le cas où l'ensemble est vide correspond aux programmes qui divergent (ou bouclent).
- Le fait que les multi-ensembles puissent avoir plusieurs éléments correspond techniquement à la possibilité de "contracter à droite" en logique classique. Cependant, on voit facilement syntaxiquement que notre terme M ne peut que diverger ou rendre un entier, et pas simultanément plusieurs entiers. On a une propriété d'*unicité des valeurs*.

Du point de vue des réseaux de preuves de la logique linéaire, cela correspond à une propriété de connexité, qui serait perdue par exemple si on introduisait une composition parallèle correspondant à la règle MIX de la logique linéaire, comme dans 4.7.7. Il est d'ailleurs facile d'étendre PCF classique avec une construction MIX qui apparaît comme une sorte de composition parallèle ou de construction de non-déterminisme *must* comme dans [DK00]. Bien sûr, avec cette construction, la proprété d'unicité des valeurs est perdue.

39. Qu'on remplace avantageusement par des !-coalgèbres ou des ?-algèbres ; on choisit ici d'utiliser des !-coalgèbres.
40. En particulier, on interprètera aussi facilement une version non déterministe de PCF classique).

Modèle des ensembles pondérés. Pour rendre compte sémantiquement de cette unicité des valeurs, j'ai construit un nouveau modèle de la logique linéaire qui est une variation sur **Rel** où les ensembles sont munis d'une fonction à valeurs dans $\mathbb{Z}$, les points "connexes" étant ceux qui sont envoyés sur 1. Tous les éléments apparaissant dans l'interprétation d'un terme de PCF classique (sans MIX) sont envoyés sur 1. Or il se trouve que les éléments de $?\mathbf{N}_0$ qui sont envoyés sur 1 sont les multi-ensembles ayant exactement 1 élément. On a ainsi une contrepartie sémantique du fait que dans $?\mathbf{N}_0$, les éléments qui comptent vraiment sont les entiers ordinaires, tous les autres (multi-ensembles de taille quelconque) étant là pour permettre l'interprétation des termes non clos.

Ce modèle est extrêmement simple et naturel et est basé sur le fait bien connu que, en logique linéaire multiplicative sans constantes, un réseau de preuve correct avec une unique conclusion a exactement un connecteur $⅋$ de plus que de connecteurs $\otimes$. Il généralise cette propriété aux exponentielles. C'est une belle illustration de l'apport de la logique linéaire à la compréhension de la sémantique de primitives de programmation comme call/cc. Ce travail sur PCF classique, mené en grande partie avec mon doctorant Shahin Amini, fait l'objet de [AE15].

4.11 Call by push value et logique linéaire (depuis l'automne 2014)

Ce travail, commencé fin 2014, vient d'une volonté de généraliser l'observation faite dans le cadre de PCF probabiliste que le type de base des entiers peut être traité en appel par valeur, dans un cadre qui est globalement par nom. J'ai d'ailleurs intégré cette observation dans mon cours de M2 MPRI dès l'automne 2014, dans le cadre de la sémantique générale de PCF.

J'ai ainsi retrouvé les idées principales de CBPV de Paul Levy dont je connaissais l'existence mais que je n'avais jamais vraiment intégré. Ma présentation semi-polarisée 4.11.1 semble plus simple que celle de Levy, et offre une connection directe avec la logique linéaire et des perspectives dans la direction de la logique linéaire différentielle. Elle se présente comme un formalisme intermédiaire entre le lambda-calcul et les réseaux de la logique linéaire.

Ce travail s'enrichit également du souhait d'étendre CBPV au calcul classique dans l'esprit de 4.10.1, ce qui me mènera à introduire un CBPV polarisé 4.11.2 qui, au fond, est plus proche du CBPV originel de Levy même si ce dernier n'avair pas la composante classique.

4.11.1 CBPV semi-polarisé

La logique linéaire est un raffinement de la logique intuitionniste (et de la logique classique). On admet généralement qu'il y a deux façons de représenter la logique intuitionniste dans la logique linéaire :

- la représentation CBN, dans laquelle on traduit $A \Rightarrow B$ par $!A' \multimap B'$ (où A' est le traduit de A et de même pour B)
- la représentation CBV où $A \Rightarrow B$ est traduit par $!(A' \multimap B')$.

Cette terminologie est due au fait que, si on applique au traduit des termes la réduction "standard" de la logique linéaire qui consiste à ne jamais réduire à l'intérieur des boîtes exponentielles, on obtient dans le premier cas une réduction CBN des lambda-termes et dans le second, une réduction CBV.

On peut comprendre le *call by push value* (CBPV) de Paul Levy comme une façon simple et naturelle de dépasser cette dichotomie en introduisant dans le lambda-calcul le concept de boîte exponentielle. Il s'agit donc d'une *généralisation* du lambda-calcul.

CBPV simplement typé minimal. Si, dans un premier temps, on ne regarde que la calcul CBPV minimal (correspondant au lambda-calcul simplement typé dans lequel le seul constructeur de types et $\Rightarrow$), on a deux classes de types en CBPV :

— les *types positifs* $\varphi = !\sigma$ (où σ est un type général)

— et les *types gnéréraux* qui sont les types positifs φ, et les types de la forme $\varphi \multimap \sigma$ où σ est un type général et φ est un type positif.

Les notations sont empruntées à la logique linéaire. Cette classification des types explique la terminologie "semi-polarisé" : on a des types positifs (polarisés donc et qui seront interprétés par des objets munis d'une structure de !-coalgèbre) et des types généraux (non polarisés, interprétés par des objets sans structure supplémentaire). En contrast avec 4.11.2 où on aura des types positifs d'une part, et négatifs de l'autre.

Les contextes de typage n'associent que des types positifs aux variables, c'est absolument essentiel.

Au niveau des termes, on trouve les variables, l'abstraction $\lambda x^{\varphi}\, t$ d'un terme t par rapport à une variable x de type positif φ (si t est de type σ, alors cette abstraction est de type $\varphi \multimap \sigma$), l'application *linéaire* $\langle s\rangle\, t$ d'un terme s de type $\varphi \multimap \sigma$ au terme t de type φ, qui est de type σ, ainsi que deux opérations supplémentaires directement héritée de la logique linéaire (bien sûr, Paul Levy ne voit sûrement pas les choses de cette façon !) :

— la *promotion*, qui transforme un terme t de type général σ en un terme $t^!$ de type $!\sigma$ (je rappelle que le contexte associe des types positifs à toutes les variables)

— la *déréliction*, qui transfome un terme t de type $!\sigma$ en un terme $\mathsf{der}(t)$ de type σ.

Contrairement à l'approche originelle de Levy dans laquelle tous les termes de type positif sont des "valeurs" (c'est-à-dire, sont duplicables et effaçables), dans notre approche, seuls les termes de type positifs qui sont de la forme $s^!$ sont des valeurs (il y a des termes de type positif qui ne sont pas de cette forme, par exemple $\langle \lambda x^{\varphi}\, x\rangle\, y$). La β-réduction s'écrit alors

$$\langle \lambda x^{\varphi}\, s\rangle\, u \rightsquigarrow s[u/x] \quad \text{à condition que } u \text{ soit une valeur}$$

et le calcul est muni d'une règle de réduction supplémentaire

$$\mathsf{der}(t^!) \rightsquigarrow t$$

qui correspond à la réduction "déréliction/boîte" de la logique linéaire. Ici "être une valeur" signifie simplement "être de la forme $t^!$" mais on verra bientôt que pour des calculs plus riches, cette notion devient plus riche également.

La restriction sur la β-réduction impose de réduire l'argument t (de type positif φ) d'une application de la forme $\langle \lambda x^{\varphi}\, s\rangle\, t$ jusqu'à l'obtention d'une valeur u avant de pouvoir procéder à la réduction.

Cette présentation a l'avantage

— de grandement simplifier celle de Levy,

— de rapprocher autant que possible le CBPV des calculs habituels

— de fournir une factorisation naturelle des deux traductions CBN et CBV du lambda-calcul dans la logique linéaire à travers CBPV.

Étant donné un modèle $\mathcal{L}$ de la logique linéaire, on interprète alors les types et le termes de façon immédiate (les notations choisies ne laissent pas d'ambiguïté), les types positifs étant interprétés par des objets de $\mathcal{L}^!$, c'est-à-dire des !-coalgèbres, et les types généraux, par des objets de $\mathcal{L}$. La remarque essentielle est que tout objet de $\mathcal{L}^!$ est un objet de $\mathcal{L}$ (avec une structure supplémentaire)

et tout morphisme de $\mathcal{L}^!$ est un morphisme de $\mathcal{L}$ qui doit en plus respecter cette structure. Un terme de type positif est interprété par un morphisme de $\mathcal{L}$ qui ne respecte pas forcément cette structure (il n'est pas forcément un morphisme de !-coalgèbres) : il est facile de prouver qu'il la respecte lorsque c'est une valeur, et dans ce cas, il est sémantiquement effaçable et duplicable.

Types plus riches, points fixes. Cette approche s'étend à des calculs beaucoup plus expressifs, englobant largement PCF et permettant de définir des types de données récursifs (entiers paresseux, listes, arbres de toutes sortes etc). L'opérateur "!" sur les types peut alors être utilisé pour imposer un comportement "paresseux".

On peut introduire des types récursifs positifs par les constructions suivantes :

— si ζ est une variable de type, alors ζ est un type positif

— si σ est un type alors $!\sigma$ est un type positif

— si φ et ψ sont des types positifs alors $\varphi \oplus \psi$ et $\varphi \otimes \psi$ sont des types positifs (type somme et type produit)

— si φ est un type positif et ζ est une variable de type, alors $\mathsf{rec}\,\zeta \cdot \varphi$ est un type positif.

et, pour ce qui est des types généraux :

— si φ est un type positif alors φ est un type général

— et si φ est un type positif et σ un type général, alors $\varphi \multimap \sigma$ est un type général.

On ne donne pas de détails sur la syntaxe des termes qui reprend les constructions du paragraphe précédent en ajoutant des constructions naturelles pour manipuler sommes et produits, ainsi qu'un opérateur pour définir des fonctions récursives, donnant accès à la Turing-complétude.

Par exemple, $0 = \mathsf{rec}\,\zeta{\cdot}\zeta$ représente le type "vide". Let type $\iota = \mathsf{rec}\,\zeta{\cdot}(!0 \oplus \zeta)$ représente le type des entiers ordinaires (on utilise !0 pour représenter un type "unité" qui contient effectivement la valeur $\Omega^!$ où Ω est un terme clos de type 0 qui boucle [41]). On obtient le type des streams d'entiers (listes paresseuses infines d'entiers) comme $\mathsf{rec}\,\zeta \cdot (\iota \otimes !\zeta)$. Une *valeur* de ce type est alors un couple $(n, s^!)$ où n est une valeur de type ι (un entier) et s est un terme de type stream, qui n'est pas forcément une valeur mais qui s'évaluera peut-être (le jour où on y accèdera au moyen de la construction $\mathsf{der}(\_)$ du langage) en un stream.

C'est donc un formalisme beaucoup plus riche que PCF (on a beaucoup plus de types de données), qui ressemble un peu au FPC de Fiore et Plotkin [FP94] avec la différence qu'ici il y a des constructions paresseuses de types donnant accès à des objets potentiellement infinis (streams par exemple) sans parler bien sûr de l'aspect CBPV évidemment absent de FPC.

Dans [Ehr15], je présente ce calcul, avec une sémantique opérationnelle faible (sans réduction à l'intérier des boîtes, ni sous les λ), je montre que les constructions de types utilisées sont bien compatibles avec la structure de !-coalgèbre que l'interprétation des types positifs doit avoir, je développe un modèle particulier basé sur la sémantique de la logique linéaire de Scott 4.7.3 pour lequel je prouve l'adéquation : la réduction d'un terme clos de type positif dont la sémantique et non vide termine.

4.11.2 CBPV polarisé

Je conclus cet article en introduisant une variante de ce calcul qui, cette fois, est complètement polarisée : on a

41. En fait tous les termes clos de type 0 bouclent.

— des types positifs ζ (variable de type), $!\sigma$ (où σ est un type négatif), $\varphi \oplus \psi$, $\varphi \otimes \psi$ et $\mathsf{rec}\,\zeta \cdot \varphi$ où φ et ψ sont des types positifs
— et des types négatifs $?\varphi$ et $\varphi \multimap \sigma$ où φ est positif et σ est négatif.

De cette manière, les règles structurelles deviennent aussi possibles à droite des séquents de typage, et on obtient un calcul *classique* (avec une représentation directe de call/cc) dont la syntaxe se base sur un calcul introduit dans [CH00] et développé dans [CM10] pour représenter les preuves du calcul des séquents classique. D'un point de vue purement logique, le calcul obtenu est un système de notations pour une version bilatérale[42] de la logique linéaire polarisée (LLP) d'Olivier Laurent (voir sa thèse, ou [LR03]). Je montre qu'on peut y representer le calcul CBPV de 4.11.1 et la thèse de Shahin Amini contiendra aussi des représentations des versions classique CBN (celle de 4.10.1) et CBV de PCF (dont je n'ai pas parlé) dans ce calcul CBPV semi-polarisé.

Cette traduction des PCF classiques factorise leur sémantique dénotationnelle dans les modèles de la logique linéaire, mais ce n'est pas le cas la traduction du FPC semi-polarisé de 4.11.1. Cela se manifeste très bien par exemple quand on doit encoder les couples (constructions de type $\otimes$) de CBPV semi-polarisé en CBPV polarisé : on a une traduction "gauche" et une traduction "droite" correspondant à deux séquentialisations différentes. C'est un effet déjà observé par Olivier Laurent que la discipline strictement polarisée de LLP impose une séquentialisation stricte des preuves/programmes. La version semi-polarisée laisse plus de liberté sur l'ordre d'évaluation.

Le cadre catégorique commun dans lequel les deux versions de CBPV s'interprètent (modèles de la logique linéaire) laisse entrevoir de nouvelles perspectives de recherche excitantes sur la sémantique catégorique de la séquentialité.

5 Publications

Les références ci-dessous se rapportent à la bibliographie située à la fin de ce rapport. Mes articles récents sont pour la plupart accessibles à partir de ma page web :

`http://www.irif.univ-paris-diderot.fr/~ehrhard/`

5.1 Publications dans des revues à comité de lecture

Parues

[Ehr93] Thomas Ehrhard. Hypercoherences: a strongly stable model of linear logic. *Mathematical Structures in Computer Science*, 3:365–385, 1993

[BE93] Antonio Bucciarelli and Thomas Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113:273–291, 1993

[BE94] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality in an extensional framework. *Information and Computation*, 110(2):265–296, 1994

[Ehr96] Thomas Ehrhard. Projecting sequential algorithms on strongly stable functions. *Annals of Pure and Applied Logic*, 77:201–244, 1996

[Ehr99] Thomas Ehrhard. A relative definability result for strongly stable functions and some corollaries. *Information and Computation*, 152:111–137, 1999

[Ehr00] Thomas Ehrhard. Parallel and serial hypercoherences. *Theoretical Computer Science*, 247:39–81, 2000

42. Avec des formules à gauche et à droite du signe "$\vdash$".

[BE00] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics in multiplicative-additive linear logic. *Annals of Pure and Applied Logic*, 102(3):247–282, 2000
[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
[Ehr04] Thomas Ehrhard. A completeness theorem for symmetric product phase spaces. *Journal of Symbolic Logic*, 69(2):340–370, 2004
[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, 2003
[Ehr02] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12:579–623, 2002
[Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005
[ER06c] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006
[ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403(2-3):347–372, 2008
[EL10b] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. *Information and Computation*, 208(6):606–633, 2010
[EL10a] Thomas Ehrhard and Olivier Laurent. Acyclic Solos and Differential Interaction Nets. *Logical Methods in Computer Science*, 6(3), 2010
[DE11] Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 152(1):111–137, 2011
[BEM12] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational semantics of parallelism and non-determinism in a functional setting. *Annals of Pure and Applied Logic*, 163(7):918–934, 2012
[Ehr12b] Thomas Ehrhard. The Scott model of Linear Logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424:20–45, 2012
[BCEM12] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full Abstraction for the Resource Lambda Calculus with Tests, through Taylor Expansion. *Logical Methods in Computer Science*, 8(4), 2012
[BET12] Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 53, 2012
[Ehr11] Thomas Ehrhard. A model-oriented introduction to differential linear logic. Submitted for publication, 2011

En révision

Le papier suivant a reçu un avis favorable pour la publication, mais je n'ai pas trouvé le temps de faire les corrections demandées...

[Ehr09] Thomas Ehrhard. On finiteness spaces and extensional presheaves over the category of polynomials. *Journal of Pure and Applied Algebra*, 2009. To appear

5.2 Articles soumis à des revues à comité de lecture ou à des conférences internationales avec comité de lecture

[EJ13] Thomas Ehrhard and Ying Jiang. CCS for Trees. Submitted for publication, 2013

[ETP15] Thomas Ehrhard, Christine Tasson, and Michele Pagani. Full Abstraction for Probabilistic PCF. Technical report, Preuves, Programmes et Systèmes, 2015. Submitted for publication to a journal

[Ehr15] Thomas Ehrhard. A Call-By-Push-Value FPC and its interpretation in Linear Logic. Technical report, Preuves, Programmes et Systèmes, 2015. Submitted for publication

5.3 Publications dans des actes de conférences internationales

[CE87] Thierry Coquand and Thomas Ehrhard. An equational presentation of higher order logic. In *Proceedings of Category Theory in Computer Science 1987*, number 283 in Lecture Notes in Computer Science. Springer-Verlag, 1987

[Ehr89a] Thomas Ehrhard. A categorical semantics of constructions. In *Proceedings of the 4th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1989

[Ehr89b] Thomas Ehrhard. Dictoses. In *Proceedings of Category Theory in Computer Science 1989*, number 389 in Lecture Notes in Computer Science. Springer-Verlag, 1989

[EM91] Thomas Ehrhard and Pasquale Malacaria. Stone duality for stable functions. In *Category Theory in Computer Science*, volume 530 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991

[BE91b] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality and strong stability. In *Proceedings of the sixth Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1991

[BE91a] Antonio Bucciarelli and Thomas Ehrhard. Extensional embedding of a strongly stable model of PCF. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, number 510 in Lecture Notes in Computer Science, pages 35–47. Springer-Verlag, 1991

[CE94] Loïc Colson and Thomas Ehrhard. On strong stability and higher-order sequentiality. In zzzzLICS'94 [LI94]

[BDER97] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Believe it or not, AJM's games model is a model of classical linear logic. In *Proceedings of the twelvth Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1997

[BDER98] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Timeless games. In *Proceedings of the Conference for Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, pages 56–77, Aahrus, Denmark, 1998. European Association for Computer Science Logic, Springer-Verlag

[BE99] Nuno Barreiro and Thomas Ehrhard. Quantitative semantics revisited (extended abstract). In *Proceedings of the fourth Typed Lambda-Calculi and Applications conference*, volume 1581 of *Lecture Notes in Computer Science*, pages 40–53. Springer-Verlag, 1999

[EL06a] Thomas Ehrhard and Olivier Laurent. Embedding the finitary pi-calculus in differential interaction nets. In *Proceedings of the Higher Order Rewriting workshop (HOR 2006)*, 2006. Electronic publication

[ER06a] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe,

and John V. Tucker, editors, *Logical Approaches to Computational Barriers*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer-Verlag, 2006. Long version available on `http://www.pps.univ-paris-diderot.fr/ ehrhard/`

[EL07] Thomas Ehrhard and Olivier Laurent. Interpreting a Finitary Pi-calculus in Differential Interaction Nets. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2007

[BEM07] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Not enough points is enough. In *Proceedings of the 21st Annual Conference of the European Association for Computer Science Logic (CSL'07)*, Lecture Notes in Computer Science. Springer-Verlag, September 2007

[BEM09] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A Relational Model of a Parallel and Non-Deterministic lambda-calculus. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 5407 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2009

[Ehr10] Thomas Ehrhard. A finiteness structure on resource terms. In *LICS*, pages 402–410. IEEE Computer Society, 2010

[CES10] Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. Resource Combinatory Algebras. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 233–245. Springer-Verlag, 2010

[ECS10] Thomas Ehrhard, Alberto Carraro, and Antonino Salibra. Exponentials with Infinite Multiplicities. In *Computer Science Logic, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2010

[BEM10] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Categorical Models for Simply Typed Resource Calculi. In *Proceedings of the Twenty-Sixth Conference on the Mathematical Foundations of Programming Semantics - MFPS XXVI*, volume 265 of *Electronic Notes in Theoretical Computer Science*, pages 213–230. Elsevier, 2010

[BCEM11] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full Abstraction for Resource Calculus with Tests. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011

[EPT11] Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherent spaces. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 87–96. IEEE Computer Society, 2011

[Ehr12a] Thomas Ehrhard. Collapsing non-idempotent intersection types. In *Computer Science Logic, 21nd Annual Conference of the EACSL, CSL 2012, September, 2012, Fontaineblau, France, Proceedings*, volume 16 of *LIPIcs*, pages 259–273. Schloss Dagstuhl - Leibniz Zentrum fuer Informatik, 2012

[ETP14] Thomas Ehrhard, Christine Tasson, and Michele Pagani. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In Suresh Jagannathan and Peter Sewell, editors, *POPL*, pages 309–320. ACM, 2014

[Ehr14] Thomas Ehrhard. A new correctness criterion for MLL proof nets. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, page 38. ACM, 2014

[AE15] Shahin Amini and Thomas Ehrhard. Classical PCF, Linear Logic and the MIX rule. 2015.

Accepted at CSL 2015

Sans rapporteurs

[Ehr95] Thomas Ehrhard. Hypercoherences: a strongly stable model of linear logic. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes Series*, pages 83–108. Cambridge University Press, 1995
qui est paru précédemment dans un journal avec comité de lecture ([Ehr93]).

5.4 Article non soumis et non publié

[BE97] Nuno Barreiro and Thomas Ehrhard. Anatomy of an extensional collapse. Manuscript, December 1997

5.5 Rapports de recherche

[EL06b] Thomas Ehrhard and Olivier Laurent. On differential interaction nets and the pi-calculus. Technical Report hal-00096280, CCSD-HAL, 2006
[BCES09] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. On linear information systems. Technical report, Preuves, Programmes et Systèmes, 2009. Presented at the Workshop LINEARITY'09
[Ehr07] Thomas Ehrhard. On finiteness spaces and extensional presheaves over the Lawvere theory of polynomials. Technical report, CCSD-HAL, 2007

Références

[AE15] Shahin Amini and Thomas Ehrhard. Classical PCF, Linear Logic and the MIX rule. 2015. Accepted at CSL 2015.

[Bar79] Michael Barr. $*$-*autonomous categories*. Number 752 in Lecture Notes in Mathematics. Springer-Verlag, 1979.

[BCEM11] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full Abstraction for Resource Calculus with Tests. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[BCEM12] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full Abstraction for the Resource Lambda Calculus with Tests, through Taylor Expansion. *Logical Methods in Computer Science*, 8(4), 2012.

[BCES09] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. On linear information systems. Technical report, Preuves, Programmes et Systèmes, 2009. Presented at the Workshop LINEARITY'09.

[BCL99] Gérard Boudol, Pierre-Louis Curien, and Carolina Lavatelli. A semantics for lambda calculi with resource. *Mathematical Structures in Computer Science*, 9(4) :437–482, 1999.

[BDER97] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Believe it or not, AJM's games model is a model of classical linear logic. In *Proceedings of the twelvth Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1997.

[BDER98] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Timeless games. In *Proceedings of the Conference for Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, pages 56–77, Aahrus, Denmark, 1998. European Association for Computer Science Logic, Springer-Verlag.

[BE91a] Antonio Bucciarelli and Thomas Ehrhard. Extensional embedding of a strongly stable model of PCF. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, number 510 in Lecture Notes in Computer Science, pages 35–47. Springer-Verlag, 1991.

[BE91b] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality and strong stability. In *Proceedings of the sixth Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1991.

[BE93] Antonio Bucciarelli and Thomas Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113 :273–291, 1993.

[BE94] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality in an extensional framework. *Information and Computation*, 110(2) :265–296, 1994.

[BE97] Nuno Barreiro and Thomas Ehrhard. Anatomy of an extensional collapse. Manuscript, December 1997.

[BE99] Nuno Barreiro and Thomas Ehrhard. Quantitative semantics revisited (extended abstract). In *Proceedings of the fourth Typed Lambda-Calculi and Applications conference*, volume 1581 of *Lecture Notes in Computer Science*, pages 40–53. Springer-Verlag, 1999.

[BE00] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics in multiplicative-additive linear logic. *Annals of Pure and Applied Logic*, 102(3) :247–282, 2000.

[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.

[BEM07] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Not enough points is enough. In *Proceedings of the 21st Annual Conference of the European Association for Computer Science Logic (CSL'07)*, Lecture Notes in Computer Science. Springer-Verlag, September 2007.

[BEM09] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A Relational Model of a Parallel and Non-Deterministic lambda-calculus. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 5407 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2009.

[BEM10] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Categorical Models for Simply Typed Resource Calculi. In *Proceedings of the Twenty-Sixth Conference on the Mathematical Foundations of Programming Semantics - MFPS XXVI*, volume 265 of *Electronic Notes in Theoretical Computer Science*, pages 213–230. Elsevier, 2010.

[BEM12] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational semantics of parallelism and non-determinism in a functional setting. *Annals of Pure and Applied Logic*, 163(7) :918–934, 2012.

[Ber78] Gérard Berry. Stable models of typed lambda-calculi. In *Proceedings of the 5th International Colloquium on Automata, Languages and Programming*, number 62 in Lecture Notes in Computer Science. Springer-Verlag, 1978.

[BET12] Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 53, 2012.

[Bou93] Gérard Boudol. The lambda-calculus with multiplicities. Technical Report 2025, INRIA Sophia Antipolis, 1993.

[BR13] Valentin Blot and Colin Riba. On bar recursion and choice in a classical setting. In Chung chieh Shan, editor, *APLAS*, volume 8301 of *Lecture Notes in Computer Science*, pages 349–364. Springer-Verlag, 2013.

[CDG+07] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on : `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[CE87] Thierry Coquand and Thomas Ehrhard. An equational presentation of higher order logic. In *Proceedings of Category Theory in Computer Science 1987*, number 283 in Lecture Notes in Computer Science. Springer-Verlag, 1987.

[CE94] Loïc Colson and Thomas Ehrhard. On strong stability and higher-order sequentiality. In LICS'94 [LI94].

[CES10] Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. Resource Combinatory Algebras. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 233–245. Springer-Verlag, 2010.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In Martin Odersky and Philip Wadler, editors, *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, pages 233–243. ACM, 2000.

[CM10] Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The Duality of Computation under Focus. In Cristian S. Calude and Vladimiro Sassone, editors, *Theoretical Computer Science - 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 165–181. Springer, 2010.

[DE11] Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 152(1) :111–137, 2011.

[DH00] Vincent Danos and Russell Harmer. Probabilistic game semantics. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2000.

[DK00] Vincent Danos and Jean-Louis Krivine. Disjunctive tautologies as synchronisation schemes. In Peter Clote and Helmut Schwichtenberg, editors, *CSL*, volume 1862 of *Lecture Notes in Computer Science*, pages 292–301. Springer-Verlag, 2000.

[ECS10] Thomas Ehrhard, Alberto Carraro, and Antonino Salibra. Exponentials with Infinite Multiplicities. In *Computer Science Logic, CSL 2010, 19th Annual Conference of the*

EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings, volume 6247 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2010.

[EGRS04] Thomas Ehrhard, Jean-Yves Girard, Paul Ruet, and Philip Scott, editors. *Linear Logic in Computer Science*, volume 316 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 2004.

[Ehr89a] Thomas Ehrhard. A categorical semantics of constructions. In *Proceedings of the 4th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1989.

[Ehr89b] Thomas Ehrhard. Dictoses. In *Proceedings of Category Theory in Computer Science 1989*, number 389 in Lecture Notes in Computer Science. Springer-Verlag, 1989.

[Ehr93] Thomas Ehrhard. Hypercoherences : a strongly stable model of linear logic. *Mathematical Structures in Computer Science*, 3 :365–385, 1993.

[Ehr95] Thomas Ehrhard. Hypercoherences : a strongly stable model of linear logic. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes Series*, pages 83–108. Cambridge University Press, 1995.

[Ehr96] Thomas Ehrhard. Projecting sequential algorithms on strongly stable functions. *Annals of Pure and Applied Logic*, 77 :201–244, 1996.

[Ehr99] Thomas Ehrhard. A relative definability result for strongly stable functions and some corollaries. *Information and Computation*, 152 :111–137, 1999.

[Ehr00] Thomas Ehrhard. Parallel and serial hypercoherences. *Theoretical Computer Science*, 247 :39–81, 2000.

[Ehr02] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12 :579–623, 2002.

[Ehr04] Thomas Ehrhard. A completeness theorem for symmetric product phase spaces. *Journal of Symbolic Logic*, 69(2) :340–370, 2004.

[Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4) :615–646, 2005.

[Ehr07] Thomas Ehrhard. On finiteness spaces and extensional presheaves over the Lawvere theory of polynomials. Technical report, CCSD-HAL, 2007.

[Ehr09] Thomas Ehrhard. On finiteness spaces and extensional presheaves over the category of polynomials. *Journal of Pure and Applied Algebra*, 2009. To appear.

[Ehr10] Thomas Ehrhard. A finiteness structure on resource terms. In *LICS*, pages 402–410. IEEE Computer Society, 2010.

[Ehr11] Thomas Ehrhard. A model-oriented introduction to differential linear logic. Submitted for publication, 2011.

[Ehr12a] Thomas Ehrhard. Collapsing non-idempotent intersection types. In *Computer Science Logic, 21nd Annual Conference of the EACSL, CSL 2012, September, 2012, Fontainebleau, France, Proceedings*, volume 16 of *LIPIcs*, pages 259–273. Schloss Dagstuhl - Leibniz Zentrum fuer Informatik, 2012.

[Ehr12b] Thomas Ehrhard. The Scott model of Linear Logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424 :20–45, 2012.

[Ehr14] Thomas Ehrhard. A new correctness criterion for MLL proof nets. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, page 38. ACM, 2014.

[Ehr15] Thomas Ehrhard. A Call-By-Push-Value FPC and its interpretation in Linear Logic. Technical report, Preuves, Programmes et Systèmes, 2015. Submitted for publication.

[EJ13] Thomas Ehrhard and Ying Jiang. CCS for Trees. Submitted for publication, 2013.

[EL06a] Thomas Ehrhard and Olivier Laurent. Embedding the finitary pi-calculus in differential interaction nets. In *Proceedings of the Higher Order Rewriting workshop (HOR 2006)*, 2006. Electronic publication.

[EL06b] Thomas Ehrhard and Olivier Laurent. On differential interaction nets and the pi-calculus. Technical Report hal-00096280, CCSD-HAL, 2006.

[EL07] Thomas Ehrhard and Olivier Laurent. Interpreting a Finitary Pi-calculus in Differential Interaction Nets. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2007.

[EL10a] Thomas Ehrhard and Olivier Laurent. Acyclic Solos and Differential Interaction Nets. *Logical Methods in Computer Science*, 6(3), 2010.

[EL10b] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. *Information and Computation*, 208(6) :606–633, 2010.

[EM91] Thomas Ehrhard and Pasquale Malacaria. Stone duality for stable functions. In *Category Theory in Computer Science*, volume 530 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[EPT11] Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherent spaces. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 87–96. IEEE Computer Society, 2011.

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3) :1–41, 2003.

[ER04] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. In *Proceedings of WoLLIC'04*, volume 103 of *Electronic Notes in Theoretical Computer Science*, pages 35–74. Elsevier Science, 2004.

[ER05] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. Technical report, Institut de mathématiques de Luminy, 2005. submitted to Theoretical Computer Science.

[ER06a] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Logical Approaches to Computational Barriers*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer-Verlag, 2006. Long version available on `http://www.pps.univ-paris-diderot.fr/ ehrhard/`.

[ER06b] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms. Unpublished long version, 2006.

[ER06c] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2) :166–195, 2006.

[ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403(2-3) :347–372, 2008.

[ETP14] Thomas Ehrhard, Christine Tasson, and Michele Pagani. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In Suresh Jagannathan and Peter Sewell, editors, *POPL*, pages 309–320. ACM, 2014.

[ETP15] Thomas Ehrhard, Christine Tasson, and Michele Pagani. Full Abstraction for Probabilistic PCF. Technical report, Preuves, Programmes et Systèmes, 2015. Submitted for publication to a journal.

[FM99] Maribel Fernández and Ian Mackie. A Calculus for Interaction Nets. In Gopalan Nadathur, editor, *PPDP*, volume 1702 of *Lecture Notes in Computer Science*, pages 170–187. Springer-Verlag, 1999.

[FP94] Marcelo P. Fiore and Gordon D. Plotkin. An Axiomatization of Computationally Adequate Domain Theoretic Models of FPC. In LICS'94 [LI94], pages 92–102.

[Gir86] Jean-Yves Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45 :159–192, 1986.

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

[Gir88] Jean-Yves Girard. Normal functors, power series and the λ-calculus. *Annals of Pure and Applied Logic*, 37 :129–177, 1988.

[Gir91] Jean-Yves Girard. A new constructive logic : classical logic. *Mathematical Structures in Computer Science*, 1(3) :225–296, 1991.

[Gir99] Jean-Yves Girard. On denotational completeness. *Theoretical Computer Science*, 227 :249–273, 1999.

[Gir04] Jean-Yves Girard. Between logic and quantic : a tract. In Ehrhard et al. [EGRS04], pages 346–381.

[Gri90] Timothy G. Griffin. The formulae-as-types notion of control. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages (POPL)*, pages 47–57. Association for Computing Machinery, January 1990.

[Her97] Hugo Herbelin. Games and Weak-Head Reduction for Classical PCF. In Philippe de Groote, editor, *TLCA*, volume 1210 of *Lecture Notes in Computer Science*, pages 214–230. Springer-Verlag, 1997.

[Hut93] Michael Huth. Linear Domains and Linear Maps. In Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt, editors, *MFPS*, volume 802 of *Lecture Notes in Computer Science*, pages 438–453. Springer-Verlag, 1993.

[Lef42] Solomon Lefschetz. *Algebraic topology*. Number 27 in American mathematical society colloquium publications. American Mathematical Society, 1942.

[Lev99] Paul Blain Levy. Call-by-push-value : A subsuming paradigm. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings*, volume 1581 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 1999.

[Lon98] J.R. Longley. The sequentially realizable functionals. Technical Report ECS-LFCS-98-402, LFCS, 1998. To appear in Annals of Pure and Applied Logic.

[LPV01] Cosimo Laneve, Joachim Parrow, and Björn Victor. Solo diagrams. volume 2215 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[LR03] Olivier Laurent and Laurent Regnier. About Translations of Classical Logic into Polarized Linear Logic. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, pages 11–20. IEEE Computer Society, 2003.

[Par92] Michel Parigot. $\lambda\mu$-calculus : An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning,International Conference LPAR'92, St. Petersburg, Russia, July 15-20, 1992, Proceedings*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 1992.

[Ret03] Christian Retoré. Handsome proof-nets : perfect matchings and cographs. *Theoretical Computer Science*, 294(3) :473–488, 2003.

[SW01] Davide Sangiorgi and David Walker. *The pi-calculus : a Theory of Mobile Processes*. Cambridge University Press, 2001.

[Vau05] Lionel Vaux. The differential lambda-mu calculus. *Theoretical Computer Science*, 379(1-2) :166–209, 2005.

[Win04] Glynn Winskel. Linearity and non linearity in distributed computation. In Ehrhard et al. [EGRS04].

[LI94] *9th IEEE Symposium on Logic in Computer Science (LICS 1994), Paris, France, Proceedings*. IEEE Computer Society, 1994.