Projet de Recherche PPS

(Preuves, Programmes et Systèmes)

10 novembre 2003

Introduction

PPS est un laboratoire qui fédère les énergies de chercheurs, enseignants-chercheurs et doctorants issus de cultures différentes (informatique et logique mathématique) pour travailler sur une thématique assez précise : celle des *langages de programmation* et des *systèmes distribués*, et de leurs *fondements logiques*. Notre projet de recherche est en effet fondé sur la conviction que la logique (et plus particulièrement la théorie de la démonstration), mais aussi la théorie des catégories, et d'autres théories mathématiques comme l'homologie ou l'homotopie, ou les probabilités, ont une part importante à jouer pour élucider le sens des programmes, afin de les rendre plus sûrs, et qu'inversement l'informatique, comme la physique a pu l'être et continue de l'être, peut être une source dans laquelle la logique et d'autres domaines des mathématiques peuvent puiser pour se renouveler.

Une grande part de la recherche théorique menée à PPS (celle décrite dans les trois premières sections de ce rapport) reste inspirée par la **correspondance de Curry-Howard** : c'est une découverte des années 1960 qui semblait n'avoir qu'une portée mineure, mais qui s'est révélée d'une importance capitale. Elle établit en effet une relation tout-à-fait remarquable entre les preuves et les programmes, qui transforme le λ-calcul en un puissant outil de théorie de la démonstration : les intuitions informatiques deviennent fondamentales en logique mathématique. Et il est fascinant de constater qu'il y a une réciproque : les méthodes de logique deviennent essentielles pour des problèmes importants posés par l'industrie informatique. Le tout premier étant le problème de la *correction des programmes.* Un vérificateur de preuves comme CoQ, développé à l'INRIA sur la base d'une extension de la théorie des types de Martin-Löf (le "calcul des constructions"), et qui est donc fondé sur la correspondance de Curry-Howard intuitionniste, est effectivement utilisé dans l'industrie pour prouver des programmes.

La correspondance de Curry-Howard s'applique dans plusieurs contextes : au départ à la logique intuitionniste (qui n'accepte par le tiers exclu), puis à la **logique linéaire**, qui réconcilie le caractère constructif de la logique intuitionniste (lequel garantit, par exemple, que toute preuve d'existence peut fournir un témoin) avec les symétries de la logique classique, et enfin à la logique classique.

Il s'agit donc d'un domaine du plus haut intérêt, entre logique et informatique, qui renouvelle complètement nos idées sur les démonstrations mathématiques, et qui est riche d'applications, non seulement à la preuve de programmes, mais aussi à la théorie des langages de programmation, au parallélisme, à la communication, etc.

Le rapport est structuré en 6 thèmes :

- **Thème I : Jeux et modèles de la programmation.** Où l'on verra que la distinction traditionnelle entre syntaxe et sémantique s'est bien atténuée, faisant apparaître certains modèles comme de la syntaxe épurée. Comme le titre l'indique, il y sera beaucoup question de *modèles de jeux*, devenus l'objet d'actives recherches depuis une dizaine d'années, et de *polarités*, apparues plus récemment.
- **Thème II : Théorie de la démonstration et λ-calcul.** Dans ce thème, intimement lié au précédent, l'accent est mis sur la syntaxe. Il y sera donc notamment question d'élimination des coupures (la contre-partie logique de l'évaluation des programmes) et de *réseaux de preuves* (qui sont des représentations géométriques des preuves que la logique linéaire a permis de construire), mais aussi de *dualités de calcul*, et de bon vieux *λ-calcul* (la charpente syntaxique de la logique intuitionniste et des langages de programmation fonctionnelle) *simplement typé*.
- **Thème III : Spécifications et réalisabilité.** L'ambition ici est de débusquer quel progamme se cache derrière tel ou tel théorème des mathématiques. Il fallait commencer par la logique classique, puis prendre les axiomes de la théorie des ensembles. Le principal outil pour cette étude est la *réalisabilité*, qui prend un peu le contre-pied des deux thèmes précédents en ce qu'elle ne suppose pas les formules avant les preuves, mais au contraire considère les formules comme des spécifications, ou ensembles de programmes (non typés) ayant le même comportement.
- **Thème IV : Réécriture.** Ici, l'on déborde du strict cadre de la logique pour étudier les propriétés des systèmes de réécriture et des *systèmes de réécriture d'ordre supérieur* (c'est-à-dire avec variables liées), pour lesquels le cadre des calculs de *substitutions explicites* est très utile. Un bon laboratoire d'utilisation des techniques de réécriture est celui des *isomorphismes de types* (avec des applications informatiques à la recherche de programmes en librairie). Des *méthodes axiomatiques pour la réécriture*

ont été développées. Un autre thème encore en émergence est celui de la *réécriture n-dimensionnelle*, où il ne s'agit plus de réécrire des mots ou des termes, mais aussi des surfaces, des volumes, etc. .

Thème V : Programmation. Le laboratoire s'est doté au cours des quatre années précédentes d'un véritable pôle de programmation, qui développe des produits logiciels, principalement (mais non exclusivement) autour du langage de *programmation fonctionnelle* Caml. Le laboratoire développe aussi une activité *systèmes* autour de Jaluna/Chorus, renforcée par le recrutement d'un professeur PAST, et une activité *protocoles réseaux* autour de IPv6. Près de la moitié des chercheurs et enseignants-chercheurs de PPS consacrent une part importante de leur temps à programmer, que cette activité soit au centre de leur activité de recherche, ou qu'elle constitue pour eux une activité annexe (contributions à des logiciels libres, par exemple).

– **Thème VI : Logique, Concurrence et Modélisation.** Un nouveau front de recherche s'est ouvert à PPS, autour de la modélisation de la programmation dite concurrente (calculs de processus) et mobile, qu'il s'agisse d'en rechercher les fondements logiques (nous avons une certaine expérience dans ce domaine), ou de les appliquer à une troisième science : la biologie, et plus précisément à la *modélisation des processus de la biologie moléculaire*.

Nous pensons que nos recherches théoriques sont particulièrement en pointe sur les sujets marqués en italique ci-dessus. Des chercheurs du laboratoire sont même à l'origine de certains d'entre eux (logique linéaire polarisée réalisabilité classique, substitutions explicites, réécriture axiomatique).

La responsabilité et la coordination de la rédaction de ce rapport ont été assurées par Emmanuel Chailloux (V), Vincent Danos (VI), Roberto Di Cosmo (IV), Jean-Louis Krivine (III), Olivier Laurent (II) et Paul-André Melliès (I et IV).

1 Thème I : Jeux et Modèles de la Programmation

Participants : Chantal Berline, Antonio Bucciarelli, Juliusz Chroboczek, Pierre-Louis Curien, Vincent Danos, Russ Harmer, Michel Hirshowitz, Olivier Laurent, Benjamin Leperchey, Jean-Vincent Loddo, François Maurel, Paul-André Melliès, Alexandre Miquel, Raphaël Montelatici, Vincent Padovani.

Souple et polyvalente, nourrie de mathématiques et de logique, la *sémantique dénotationnelle* est devenue en trente-cinq ans un outil canonique dans l'analyse et la modélisation des langages de programmation et des démonstrations.

Les langages bâtis sur un noyau fonctionnel de λ-calcul y sont modélisés en termes de catégories cartésiennes fermées, où les propriétés essentielles du langage apparaissent sans les détails de la syntaxe. Les constructions sont toutes *modulaires* ou *compositionelles* : la sémantique d'un programme est définie à partir de la sémantique de ses sous-programmes, et une optimisation de code est validée dès lors qu'elle remplace une procédure P par une procédure Q de même sémantique.

L'histoire de la sémantique dénotationnelle est jalonnée par trois découvertes charnières :

1. la *sémantique des domaines* (statique, extensionnelle), qui a permis d'interpréter les programmes du λ-calcul avec récursion par des fonctions croissantes et continues entre ensembles ordonnés avec limites filtrées (appelés domaines).
2. la *logique linéaire*, introduite par Girard en 1986, qui a montré que la flèche intuitionniste $A \Rightarrow B$ du λ-calcul se décompose en une flèche linéaire $A \multimap B$ et une modalité de répétition $!A$; et a dévoilé du même coup une dualité fondamentale entre Programme et Environnement, interprétée logiquement comme une négation involutive (comme la négation classique) sur les formules : $A^{\perp\perp} \cong A$.
3. la *sémantique des jeux* (dynamique, intensionnelle), qui a raffiné la sémantique des domaines en y introduisant le temps, ce qui a permis de modéliser exactement (par des résultats de complétude forte, voir plus loin) des langages fonctionnels ou impératifs avec variables locales, références, gestion des continuations, des exceptions, etc.

Ces cinq dernières années ont été marquées par un mouvement soutenu de synthèse et de rapprochement de ces trois champs. Les membres de PPS ont joué un rôle moteur dans cette convergence, en établissant des passerelles théoriques importantes. Ces travaux ont révélé certains concepts clefs au cœur de la séquentialité : erreurs, polarités, et concurrence (sections 1.1, 1.2, 1.3). Les modèles probabilistes, les jeux combinatoires avec gain, et la sémantique du polymorphisme, ainsi que diverses autres questions liées à la sémantique, n'ont pas été oubliés. (sections 1.4, 1.5, 1.6, 1.7).

1.1 Valeur d'erreur et λ-calcul : entre modèles intensionnels et extensionnels

La distinction traditionnelle entre modèles intensionnels et extensionnels du λ-calcul a été bousculée au début des années 1990 par les travaux de Cartwright et Felleisen, qui ont montré que l'ajout d'une valeur d'erreur $\mho$ à un modèle extensionnel en dévoile certains phénomènes intensionnels. Cette

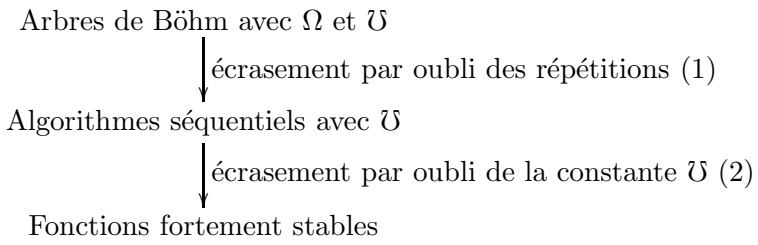
idée est approfondie à divers degrés dans plusieurs travaux de l'équipe, qu'ils étudient :

- le modèle des algorithmes séquentiels pour le décrire extensionnellement (Curien) ou pour le caractériser comme écrasement extensionnel d'autres modèles plus canoniques de la séquentialité (Melliès),
- les hiérarchies de λ-définissabilité dans les modèles extensionnels du λ-calcul (Bucciarelli, Leperchey, Padovani),
- les propriétés du sous-typage dans des modèles de réalisabilité fondés sur les jeux (Chroboczek), la ludique probabiliste (Maurel, voir section 1.4) ou le λ-calcul (Melliès et Vouillon, voir thème III).

Une formulation extensionnelle des algorithmes séquentiels. Après leur redécouverte par Cartwright et Felleisen [CF92, CCF94] en 1991, avec l'ajout majeur des valeurs d'erreur dans le modèle, puis la décomposition affine du modèle (initialement proposée en 1992 par Lamarche [Lam92, Cur93]), le modèle des algorithmes séquentiels de Berry et Curien [BC82, AC98] a connu une troisième renaissance en 2002 grâce aux travaux de Laird [Lai02], qui réhabilite (après Chroboczek) les treillis en sémantique dénotationnelle, l'élément maximum correspondant à la valeur erreur, ou encore au *démon* de la ludique de Girard (ou preuve inachevée). Techniquement, Curien montre dans [Cur03] que la catégorie des algorithmes séquentiels avec erreurs se plonge fidèlement dans la catégorie des biordres bistables de Laird. Ainsi, le modèle très concret des algorithmes séquentiels, précurseur des sémantiques de jeux, peut être décrit totalement implicitement comme un bête modèle de fonctions sur des structures (bi-)ordonnées, de surcroît particulièrement simples. Les deux points de vue se complètent pour leur mutuel enrichissement, comme dans tous les théorèmes de représentation en mathématiques.

Le modèle des algorithmes séquentiels comme écrasement extensionnel. Melliès tire des travaux de Cartwright et Felleisen une topographie renouvelée des modèles du λ-calcul, qui "factorise" la thèse dite de Longley, selon laquelle tous les langages séquentiels suffisamment expressifs devraient s'écraser extensionnellement sur la hiérarchie fortement stable de Bucciarelli et Ehrhard. Melliès montre en effet qu'une fois la notion d'erreur prise en compte, un nombre important de hiérarchies séquentielles s'écrase extensionnellement en la hiérarchie $\mathcal{H}$ des algorithmes séquentiels munie d'une valeur d'erreur $\mho$. C'est le cas en particulier (1) de la hiérarchie syntaxique des λ-termes avec constante de non-terminaison Ω et constante d'er-

reur $\mho$:



Mellliès adapte ensuite le théorème de Ehrhard discuté en section 1.3 au cadre avec erreur, pour montrer (2) que la hiérarchie $\mathcal{H}$ s'écrase sur la hiérarchie des fonctions fortement stables lorsque la valeur d'erreur $\mho$ n'est pas prise en compte dans les types de base. Ces résultats sont établis au moyen d'une notion catégorique de traduction "aller-retour" (back-and-forth) entre modèles de la logique linéaire, détaillée en [Mel02c, Mel03e].

Définissabilité et modèles extensionnels. L'ajout d'une valeur d'erreur $\top$ au domaine booléen à trois valeurs V, F et $\bot$ dans la sémantique de Scott le transforme en un treillis $\mathbb{B}$ à quatre valeurs, qu'on peut identifier au type $o \Rightarrow o \Rightarrow o$ d'un λ-calcul simplement typé muni d'une constante de non-terminaison Ω et d'une constante d'erreur $\mho$. Ce λ-calcul est appelé *PCF unaire* lorsqu'on lui rajoute la récursion Y : unaire signifie simplement que le type o a pour interprétation le domaine de Sierpinski $\{\bot, \top\}$. La traduction de $\mathbb{B}$ en $o \Rightarrow o \Rightarrow o$ (traduction de continuations, au sens informatique) permet de prendre pour valeurs booléeenes $V, F, \bot, \top$ les formes normales η-longues du type $o \Rightarrow o \Rightarrow o$:

$$V = \lambda xy.x, \qquad F = \lambda xy.y, \qquad \bot = \lambda xy.\Omega, \qquad \top = \lambda xy.\mho.$$

Nous verrons plus loin (section 1.2) que cette traduction correspond à la traduction *polarisée* des booléens en logique linéaire : $\mathbb{B} = ?(1 \oplus 1)$ — qui remplace la traduction $\mathbb{B} = 1 \oplus 1$ choisie par exemple en sémantique des espaces cohérents.

Récemment, Laird a remarqué [Lai03] que PCF unaire n'admet que deux modèles extensionnels ordonnés : le modèle de Scott et le modèle complètement adéquat. La question se posait donc de savoir si, de même, les degrés de définissabilité du modèle de Scott de PCF unaire sont réduits à deux (celui des fonctions définissables et celui du "test parallèle de convergence"). Bucciarelli, Leperchey et Padovani ont montré dans [BLP03] que c'est le contraire qui se produit : l'ordre partiel des degrés de définissabilité est non-trivial. Un problème ouvert intéressant est celui de la décidabilité de cet ordre partiel.

Leperchey travaille avec Bucciarelli sur la définissabilité relative dans les modèles extensionnels des langages de programmation fonctionnels (en ce

moment plus spécifiquement sur les fonctions fortement stables [Lep03]) et sur les applications éventuelles à la programmation (comme, par exemple, dans les travaux de Longley [Lon99]).

L'étude de la définissabilité dans les espaces de cohérence par des preuves en logique linéaire, a mené Bucciarelli et Thomas Ehrhard (Institut de Mathématiques de Luminy) á la définition d'une nouvelle classe de modèles dénotationnels de la logique linéaire [BE00, BE01].

Modèles de jeux pour le sous-typage. Le sous-typage est une caractéristique essentielle des langages de programmation, notamment des langages à objets. Malgré son importance, on n'en connaît que très peu de modèles sémantiques.

Une façon de comprendre le typage — et donc le sous-typage — est de partir de l'idée que le typage sert à éviter une certaine classe d'erreurs ; cette approche est résumée dans le slogan *well typed terms can't go wrong* (les termes bien-typés ne terminent pas en catastrophe).

Dans sa thèse [Chr01a], Chroboczek a construit un modèle de jeux d'un langage avec erreurs explicites. Étant donné un terme dans un tel langage, il est naturel de s'intéresser à la classe des contextes où il ne génère pas d'erreurs ; étant donnés deux tels termes, l'inclusion des classes de contextes associés induit un ordre $\leq$ qui, sous les bonnes conditions, coïncide avec l'ordre extensionnel. A partir d'une définition des types dans l'esprit de la sémantique des jeux (comme ensemble de positions) on démontre que tout type est un idéal pour l'ordre $\leq$. De même, une notion d'ordre sémantique est donnée sur les types, et on démontre qu'un type A est plus petit qu'un type B si et seulement si les stratégies de A sont toutes des stratégies de B. On obtient de cette manière un modèle "à la Curry" dans l'esprit de la réalisabilité (cf. thème III). Et on démontre que l'espace (ordonné) des types a les propriétés requises pour construire une interprétation du polymorphisme, des types abstraits (quantification universelle et existentielle) [Chr00] ainsi que des types récursifs [Chr01b].

Ces travaux éclaircissent la sémantique des langages à objets, mais ne débouchent malheureusement pas encore sur un modèle complètement adéquat de ces langages. Des problèmes difficiles et techniques apparaissent en effet, que Chroboczek pense résoudre un jour en tirant avantage du caractère "à la Curry" de ces modèles. Ces travaux permettent aussi de mieux comprendre les relations entre sémantique dénotationnelle, sémantique par réécriture et machines à environnement dans le cadre spécifique des langages fonctionnels avec erreurs explicites [Chr02]. Ils établissent en particulier des liens inattendus avec la sémantique d'ALGOL 60.

1.2 Logique linéaire polarisée et sémantique des jeux

L'histoire de la logique linéaire polarisée illustre bien le paradigme de Curry-Howard, et le rapprochement inexorable entre logiciens et informaticiens. L'étude par Girard de la sémantique dénotationnelle de la logique classique, et la découverte par Andréoli de mécanismes de synchronisation en logique linéaire (appelés focalisation) a permis de dégager une notion de *polarisation* qui distingue les formules *positives* et *négatives* de la logique linéaire. Le fragment polarisé de la logique linéaire (cf. thème II) est défini en restreignant l'application du produit tensoriel $\otimes$ et de la somme $\oplus$ (et leurs éléments neutres 0 et 1) aux formules positives P, et dualement l'application du produit parallèle $\parr$ et du produit & (et leurs éléments neutres $\top$ et $\bot$) aux formules négatives N :

$$P ::= \ P \oplus P \mid P \otimes P \mid 0 \mid 1 \qquad N ::= \ N \parr N \mid N \& N \mid \top \mid \bot.$$

Ajoutée à ces constructions, la modalité *exponentielle* ! et une modalité dite de *décalage* $\downarrow$ permettent de transformer une formule négative N en formule positive, et dualement, les modalités ? et $\uparrow$ transforment une formule positive P en formule négative :

$$P ::= !N \mid \ \downarrow N \qquad N ::= ?P \mid \ \uparrow P.$$

On dégage ainsi une logique linéaire polarisée (LLP), version "saine" de la logique linéaire, et cadre confortable pour analyser les diverses extensions de Curry-Howard à la logique classique, le $\lambda\mu$-calcul de Parigot en particulier.

La prise en compte des polarités en sémantique des jeux permet une analyse logique de la dualité entre Joueur et Opposant : les formules positives sont interprétées par des jeux "positifs" où *Joueur* commence, et (dualement) les formules négatives par des jeux "négatifs" où *Opposant* commence. Point intéressant : les modalités exponentielle ! et décalage $\downarrow$ changent la polarité d'une formule (ou d'un jeu) en rajoutant un coup initial Joueur à tout jeu négatif.

Selinger a introduit récemment une notion de modèle catégorique pour le $\lambda\mu$-calcul : les *catégories de contrôle* (voir thème II) pour lesquelles il démontre un *théorème de représentation* inspiré par les travaux de Hofmann et Streicher : toute catégorie de contrôle provient d'une catégorie de continuations, c'est-à-dire d'une catégorie cartésienne et co-cartésienne disposant d'un objet o exponentiable.

La logique linéaire polarisée rejoint ainsi à la notion bien connue de *continuation* chez les informaticiens — notion qu'elle revisite en lui offrant un faisceau d'outils nouveaux : dualité classique, réseaux de démonstration, géométrie de l'interaction, sémantique des jeux, logique du second ordre. Un exemple parmi d'autres, que nous signalions en section 1.1 : la traduction

habituelle des booléens $\mathbb{B} = o \Rightarrow o \Rightarrow o$ dans un modèle de continuations, s'écrit plus clairement $\mathbb{B} = ?(1 \oplus 1)$ en logique linéaire polarisée. On retrouve là l'interprétation habituelle des booléens en sémantique des jeux : la formule $1 \oplus 1$ se lit : V ou F, tandis que la modalité ? se lit : question initiale de l'Opposant.

La logique LLP a été introduite par Laurent à partir des travaux de Girard sur LC. Des liens sont apparus très vite avec les travaux de Parigot sur le $\lambda\mu$-calcul, ceux de Danos, Joinet et Schellinx sur les traductions de la logique classique en logique linéaire, et dans un registre plus syntaxique ceux de Curien et Herbelin sur la dualité de calcul (cf. thème II et les travaux de Griffin). Plusieurs membres de l'équipe se sont intéressés à la sémantique dénotationnelle de la polarisation, soit pour analyser la structure catégorique des catégories de contrôle (Laurent, Melliès), soit pour établir des résultats de complétude forte sur les modèles de jeux polarisés (Laurent, Montelatici), ou enfin pour ajouter la récursion à LLP (Montelatici).

Catégories de contrôle. Les catégories de contrôle ont été étudiées sous plusieurs angles. Laurent et Regnier [LR03] ont proposé une construction de catégories de contrôle à partir de modèles de la logique linéaire. Les espaces de corrélation [Gir91] (premier modèle dénotationnel classique) apparaissent comme le cas particulier de cette construction appliquée aux espaces cohérents.

Melliès s'est joint à Selinger [MS03] pour étudier la structure catégorique fine des modèles de contrôle et de continuation dans le cas des sémantiques de jeux. Un point de contact avec la construction de famille proposée par Abramsky et McCusker [AM97] a été établie, ainsi qu'une décomposition linéaire du théorème de représentation de Selinger. La comparaison entre ces travaux et ceux de Laurent et Regnier reste à mener.

Résultats de complétude. Laurent [Lau02b, Lau03b] étudie des modèles de jeux complets à la Hyland-Ong (HO) et à la Abramsky-Jagadeesan-Malacaria (AJM) pour la logique LLP. Un résultat de complétude forte est établi, qui énonce que l'interprétation est surjective : toute stratégie du modèle est l'interprétation d'une certaine démonstration de LLP. Parce que ces modèles de jeux permettent aussi d'interpréter la logique ILL (logique linéaire intuitionniste) il est instructif de comparer les deux interprétations. On relie par exemple le produit tensoriel intuitionniste $\odot$ et le produit tensoriel polarisé $\otimes$ par $\downarrow(A \odot B) = \downarrow A \otimes \downarrow B$. On peut aussi vouloir renforcer le théorème de complétude forte (surjectivité) et démontrer une équivalence exacte (surjectivité et injectivité) entre stratégies et démonstrations, à condition d'utiliser une syntaxe de réseaux à tranches (voir thème II).

Dans son mémoire de DEA, Montelatici [Mon01] axiomatise les propriétés d'une catégorie de continuation pour que la catégorie de contrôle associée vérifie le théorème de complétude forte pour la logique LLP. Il démontre que la catégorie des jeux d'arènes à la HO et stratégies innocentes, déjà étudiée par Laurent, vérifie cette axiomatique.

Modèles dénotationnels de la logique classique. Les modèles de jeux polarisés ont montré comment décomposer les connecteurs de la logique linéaire à l'aide d'opérateurs de décalage. Cette décomposition apparaît-elle dans d'autres types de modèles ? Des premiers pas dans cette direction sont donnés dans [Lau02b] par la définition de la notion de *catégories de contrôle linéaires* qui s'avère coïncider avec plusieurs linéarisations de systèmes syntaxiques pour la logique classique.

Sémantique du combinateur Y de point fixe. Montelatici [Mon03] a développé une extension de LLP avec point fixe, avec une notion de réseau cyclique (voir thème II) et une sémantique des jeux avec récurseur Y définie à partir des travaux sur les catégories compactes closes de Freyd, Plotkin, et Simpson. Point intéressant : le point fixe est construit exactement de la même manière dans les modèles de jeux et dans les modèles de Scott, sitôt que la monade T de soulèvement (lift) des domaines ($D \mapsto D_\perp$) est remplacée par la monade T de double décalage sur les jeux : ($A \mapsto \uparrow \downarrow A$). Différence technique importante cependant : la monade T est *commutative* sur les domaines, et seulement *forte* (et non commutative) sur les jeux. Notons ici un point théorique troublant : l'interprétation catégorique du point fixe Y en LLP nécessite un cadre plus généreux que celui de LLP : on est obligé de revenir à la catégorie cartésienne close sous-jacente, et ce faisant, à ILL. Cela indique un point limite de LLP, qui mérite d'être analysé plus avant.

1.3 Jeux asynchrones : entre concurrence et séquentialité

Malgré ses succès, la sémantique dénotationnelle reste marquée par le tournant opérationnel de Milner à la fin des années 70, lorsqu'il construit sa première théorie des calculs de processus concurrents. Milner jugeait la sémantique dénotationnelle incapable de décrire les *mécanismes d'interférence* entre processus communicants. Il opta donc pour un bricolage génial, qui consiste : (1) à formaliser un calcul de processus (CCS en 1980, puis le π-calcul en 1990), où des scénarios idéalisés de communication concurrente furent modélisés ; et (2) à lui donner une sémantique opérationnelle fondée sur l'idée de *bisimulation*. Exit la sémantique dénotationnelle !

Vingt ans plus tard, la théorie des calculs de processus reste coupée de la sémantique dénotationnelle. Cette absence de fondations dénotationnelles explique par exemple les difficultés à *typer* le π-calcul (ou ses variantes) c'est-à-dire à le structurer selon une grammaire logique similaire à celle des langages construits autour d'un noyau fonctionnel de λ-calcul (cf. thème VI). Plusieurs équipes dans le monde travaillent à combler le fossé qui sépare calculs de processus et sémantique dénotationnelle. Deux axes sont généralement privilégiés. D'un côté, des chercheurs issus de la sémantique (Fiore, Plotkin, Turi, Winskel) développent une sémantique des faisceaux et une théorie algébrique et coalgébrique des calculs de processus concurrents. De l'autre, des chercheurs issus de la concurrence (Honda, Sangiorgi, Yoshida) étudient les fragments séquentiels du π-calcul (ou variations) afin de les comparer avec des fragments de λ-calcul, et de les typer (voir cependant thème VI, section 6.3).

Nous suggérons ici un troisième angle d'attaque, fondé sur une synthèse récente entre *sémantique des jeux* et *géométrie de la concurrence*, opérée par Melliès sous le nom de sémantique *asynchrone* des jeux. Ce cadre asynchrone et interactif permet une analyse concurrente des mécanismes d'adressage du λ-calcul, qui révèle que les stratégies du λ-calcul sont *positionnelles* (elles jouent en fonction de leur position courante, pas de leur trajectoire toute entière) et *confluentes* (elles vérifient une version *interactive* du théorème de Church-Rosser). Le théorème de complétude forte (full abstraction) affirmait qu'il est possible d'*extraire* la "syntaxe" du λ-calcul de sa "sémantique". La syntaxe apparait alors comme un "produit dérivé" et imparfait de la sémantique — ce qui laisse espérer par exemple l'extraction de syntaxes concurrentes à partir d'une sémantique des jeux proprement concurrente. Au passage, le cadre asynchrone permet aussi de modéliser topologiquement (par des trous) les *interférences* des processus interactifs, dans l'esprit de la géométrie du calcul (cf. thème IV).

Mis bout à bout, ces travaux incorporent à la sémantique dénotationnelle tous les ingrédients qui ont servi à Milner entre 1975 et 1990 pour bâtir sa théorie des calculs de processus [1] :

i. systèmes de transition (sémantique des jeux),

ii. interférences entre processus communicants (jeux asynchrones),

iii. traductions par continuations du λ-calcul (traductions polarisées).

Nous espérons voir bientôt fleurir une sémantique dénotationnelle des calculs de processus concurrents — où ces calculs seront réexaminés à la lumière de notre compréhension nouvelle (concurrente et interactive) des mécanismes d'adressage du λ-calcul.

[1] Rappelons que CCS est le produit des deux premiers ingrédients, et le π-calcul des trois ingrédients mis bout-à-bout

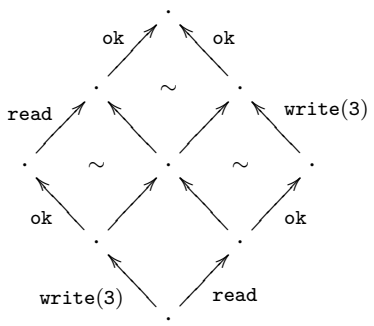
Algorithmes séquentiels et fonctions fortement stables. Étape préparatoire à la sémantique des jeux asynchrones, Melliès [Mel03d] redémontre le théorème d'écrasement de Ehrhard dans un esprit diagrammatique et concurrent. Ce théorème déjà mentionné en section 1.1 énonce que la hiérarchie des algorithmes séquentiels de Berry et Curien s'écrase extensionnellement sur la hiérarchie fortement stable de Bucciarelli et Ehrhard. Le théorème découle chez Ehrhard [Ehr97], puis chez Longley et van Oosten, d'une propriété d'*universalité* qui énonce que toute fonction fortement stable est définissable par un programme PCF appliqué à des fonctions fortement stables de type 2 [Ehr97]. Melliès donne une démonstration toute différente, en identifiant la sémantique des jeux à une sémantique concurrente par *entrelacement* (interleaving) et la sémantique fortement stable à une sémantique concurrente *véritable* (true concurrency). La démonstration établit ainsi un point de contact balbutiant mais très instructif entre sémantique des jeux, sémantique des domaines, et théorie de la concurrence.

Jeux AJM et actions de groupe. Seconde étape préparatoire à la théorie asynchrone des jeux, Melliès [Mel03a] reformule dans l'esprit de la théorie des groupes la notion d'auto-équivalence des jeux d'Abramsky, Jagadeesan et Malacaria (AJM). Les jeux sont munis d'une action de groupe à gauche (pour Joueur) et à droite (pour Opposant). On montre qu'une stratégie σ est auto-équivalente au sens AJM ssi elle est *bi-invariante*, c'est-à-dire pour tout élément $s \in \sigma$ et toute action h à droite, il existe une action g à gauche, telle que $g \cdot s \cdot h \in \sigma$.

Jeux asynchrones. Les résultats obtenus en [Mel03d] sont encourageants, mais incomplets, car les tuiles de concurrence y sont *globales* et non pas *locales* dans l'esprit de Marzukiewicz ou de la réécriture (voir thème IV). Aussi, Melliès propose en [Mel03a, Mel03b] une *sémantique des jeux asynchrones*, où l'arbre de jeu habituel est remplacé par un système de transition asynchrone — où les coups commutent donc localement. Ce cadre permet d'analyser les propriétés diagrammatiques des stratégies *innocentes* (autrement dit, définies par un arbre de Böhm) dans les jeux d'arènes de Hyland, Ong et Nickau [HO00, Nic94]. Melliès montre par des méthodes de réécriture que toute stratégie innocente σ est positionnelle. La stratégie σ définit en fait un opérateur de clôture dans le treillis des positions, qui vérifie une version interactive du théorème de Church-Rosser. On obtient de la sorte un rapprochement entre jeux d'arène de Hyland, Ong et Nickau, et jeux concurrents d'Abramsky et Melliès [AM99].

Interférence et géométrie de la concurrence On peut rajouter aux jeux asynchrones une relation d'indépendance I sur les coups, qui rend

compte des phénomènes d'*interférence* ou d'*effets de bord* dans les langages de programmation impératifs. Ainsi, dans Algol Idéalisé, on indiquera que lecture et écriture d'une même cellule mémoire interfèrent en imposant par exemple que les coups `read` et `write(3)` ne sont pas indépendants dans le jeu asynchrone interprétant la variable de type Nat :



On obtient un *trou* qui fait obstruction à la relation d'homotopie $\sim$ entre trajectoires interactives, dans l'esprit géométrique du thème IV. L'état courant du système est donné dans ce cas par la *classe d'homotopie* de la trajectoire d'interaction, plutôt que par sa simple position d'arrivée, comme c'est le cas pour les stratégies positionnelles du λ-calcul.

Arbres de Böhm abstraits. La sémantique des jeux asynchrones indique de nouvelles classes d'arbres de Böhm pour les calculs concurrents ou impératifs. Il apparait donc important de dégager une notion suffisamment abstraite et générale d'arbre de Böhm. Curien a continué de développer le formalisme des arbres de Böhm abstraits, conçu au départ comme une généralisation commune des arbres de Böhm et de leurs variantes pour PCF ou pour le $\lambda\mu$-calcul [Cur98]. Tous ces arbres ont en commun de constituer une sémantique algébrique ou "initiale" des langages correspondants, en ce sens que l'on associe à un programme sa forme normale par réduction (confluente, cf. thème IV), possiblement infinie. Récemment, Curien a proposé une syntaxe concrète pour ces arbres, qui devrait les rendre plus maniables que la présentation initiale avec pointeurs explicites à la De Bruijn ou à la Hyland-Ong. Deux exemples nouveaux de compilation injective vers les arbres de Böhm abstraits sont : les arbres de Böhm par valeur (qui semblent être une notion nouvelle) et les desseins de la ludique. Toutes ces observations seront détaillées dans [Cur].

1.4 Modèles probabilistes.

De la même façon qu'avec la valeur d'erreur, la prise en compte des probabilités en sémantique des jeux (Danos, Harmer) ou en ludique (Maurel) permet

de révéler au niveau extensionnel certains mécanismes intensionnels : en particulier la répétition par Joueur d'une même question à Opposant, invisible dans un modèle traditionnel.

Sémantique des jeux probabilistes et non déterministes. Danos et Harmer ont étendu la sémantique des jeux au cas probabiliste [DH00, DH02] afin de modéliser des programmes aléatoires comme le "randomized quicksort".

Harmer a aussi travaillé sur les jeux non-déterministes. En collaboration avec McCusker, il a généralisé au cas infini le modèle du non-déterminisme fini figurant dans sa thèse [Har99]. Ce travail a permis de simplifier également le modèle original du non-déterminisme fini. Ces travaux sont précisés en [HM].

Ludique probabiliste. Maurel développe une version probabiliste de la ludique [Gir01]. La ludique est issue d'une distillation de la logique linéaire qui prend en compte les polarités et la notion de preuve inachevée (cf. plus haut), laquelle induit une notion de convergence ou terminaison : le consensus par abandon (penser à une discussion animée, qui ne s'arrête que lorsqu'un des deux interlocuteurs est à court d'arguments). La ludique introduit des objets non typés appelés dessins, où ne restent des formules que leurs adresse, à partir desquels les formules sont reconstruites en tant que comportements, ou ensembles de formules clos par bi-orthogonalité (voir thème III).

Les dessins probabilistes de Maurel permettent de discuter plus finement de propriétés telles que la divergence, ou la réutilisation de lieux qui correspond à utiliser plusieurs fois une même formule. Ce modèle est une sorte de probabilisation du modèle des arbres de Böhm abstraits donné par Curien [Cur98]. La construction préserve l'associativité (qui correspond à l'élimination des coupures). La séparation n'est pas préservée, mais le sous-ensemble des stratégies non probabilistes (y compris avec répétition) est séparé par le reste du modèle. Ce phénomène provient de l'utilisation des coefficients : le sous-modèle des stratégies non probabilistes n'est pas séparé. Les stratégies non-probabilistes d'un type sont caractérisées géométriquement comme les points extrémaux de ce type. La formalisation d'un théorème de complétude est en cours.

1.5 Jeux combinatoires avec gain et programmation logique

Bien qu'elle joue un rôle fondamental dans la théorie classique des jeux combinatoires, la notion de gain est restée jusqu'ici quasiment absente de la

sémantique des jeux. Des travaux récents de Loddo ont établi les préliminaires d'une sémantique et d'une algorithmique de ces jeux avec gain, en les rapprochant de la programmation logique [DCL00, Lod02, Lod03].

Généralisation des jeux combinatoires. Afin de pouvoir être utilisées en sémantique des langages de programmation, les notions de *jeu* et de *gain* de la théorie des jeux combinatoires doivent être généralisées. Pour cela, le concept de gain est axiomatisé à partir de l'exemple des booléens (*gagné*, *perdu*) ou encore des entiers (qui modélisent typiquement un gain en argent) ; et les jeux combinatoires sont étendus afin d'admettre des parties *infinies* entre Joueur et Opposant.

Il faut ensuite généraliser les algorithmes de résolution de la théorie des jeux combinatoires. Loddo montre de manière axiomatique que le fameux algorithme Alpha-Bêta fonctionne sur un très large éventail de notions de gain. Il prévoit de donner une formulation à petits pas (*small steps*) de l'algorithme Alpha-Bêta, dans le style des sémantiques opérationnelles structurées (SOS). Cela rendrait le *parallélisme* de l'algorithme explicite, et fournirait le cadre théorique d'une implantation d'Alpha-Bêta pour la programmation logique (voir plus bas).

Jeux combinatoires et programmation logique. Il devient alors possible de ré-expliquer la programmation logique (*Prolog*, *DataLog*) et la programmation par contraintes (*CLP*) en termes de jeu combinatoire à deux joueurs. Ce parallèle établit un pont inattendu entre deux disciplines, qui, au delà de l'intérêt purement sémantique (et peut-être pédagogique), permet aussi d'envisager la résolution d'un but dans un programme logique, de la même façon "intelligente" qui caractérise les algorithmes qui "jouent" à des jeux de stratégie. En effet, il est possible de résoudre un but dans un programme logique par la méthode Alpha-Bêta. La méthode réalisera des simplifications du calcul bien connues en théorie des jeux mais tout à fait nouvelles et profitables en programmation logique. La théorie développée par Loddo fournit donc une base solide à l'implémentation d'un moteur de résolution de buts logiques basé sur Alpha-Bêta, qui reste à construire. D'un point de vue plus théorique, des questions nouvelles émergent par la simple volonté d'expliquer en termes de jeux les outils théoriques développés pour et autour de la programmation logique, comme la *négation* ou l'*interprétation abstraite* de programmes logiques.

1.6 Sémantique du polymorphisme

Sémantiques de jeux et polymorphisme. Les modèles de jeux ont majoritairement été étudiés dans le cadre de langages construits sur des types

de base (constantes en logique, entiers ou booléens), le traitement de langages avec variables de type étant généralement plus lourd. Dans [Lau03d], Laurent a décrit un modèle de jeux qui prend en compte les variables de types de manière simple. L'enjeu est maintenant d'étendre le modèle à la quantification du second ordre en préservant sa simplicité.

Domaines et polymorphisme. Berline et Berardi (Turin) ont étudié les modèles du système F de Girard et Reynolds et de ses extensions (F_ω, calcul des constructions, et $Type : Type$). En particulier ils ont montré dans [BB02] que le modèle du système F construit par Berardi et Barbanera était "complet pour la $\beta\eta$-équivalence", en ce sens qu'il n'égale que des termes $\beta\eta$-équivalents ; il s'agit du premier modèle vérifiant cette condition, si l'on excepte l'évident modèle des termes, et il s'agit d'un modèle simple (non catégorique) qui vit dans la sémantique continue de Scott.

Berline et Berardi ont isolé une notion abstraite, mais très naturelle, de "modèles du système F" qui n'est pas exhaustive, mais est très commode. Berline a isolé à l'intérieur de cette classe, une sous-classe plus concrète de modèles de la sémantique continue, qui contient le modèle de Barbanera et Berardi. Ces modèles ne sont pas tous complets pour la $\beta\eta$-équivalence, mais ils possèdent tous une "trame" à partir de laquelle ils peuvent être reconstruits, et modélisent des extensions variées du système F (ajout de constantes, de contraintes, etc.) [BB04].

Berline et Berardi ont remarqué que bon nombre de ces modèles étaient en fait des modèles de F_ω, et que parmi eux il y avait des modèles du calcul des constructions et de $Type : Type$. Pour rédiger l'article correspondant à ces résultats, ils ont senti le besoin de construire un cadre commun où l'on pourrait exprimer de façon précise et *maniable* ce que signifiait "être un modèle" de ces systèmes complexes. Ce point s'est révélé techniquement complexe et résiste encore.

En plus de ces travaux sur le polymorphisme, Berline a travaillé sur les modèles du lambda-calcul non typé [Ber00] et de ses extensions.

1.7 Varia

λ-définissabilité et modèles extensionnels. Bucciarelli a poursuivi ses travaux sur la λ-définissabilité dans les modèles extensionnels. Avec Leperchey, il a donné une caractérisation complète de la PCF-définissabilité relative des fonctions booleénnes "sous-séquentielles" [BL03]. Une nouvelle notion de morphisme d'hypergraphes (*timed morphism*) est à la base de la preuve de complétude. La preuve de correction reprend celle de [BM02].

Dans [BPS03], Bucciarelli, Adolfo Piperno (Université de Rome) et Ivano

Salvo (Université de Turin) prouvent que, en ce qui concerne la définissabilité (uniforme) des fonctions récursives, les types simples et les types intersection ont le même pouvoir expressif.

Dans [BS03], Bucciarelli et Antonino Salibra (Université de Venise) montrent que la classe des modèles de graphes (graph-models) du λ-calcul pur admet un plus petit modèle, c'est-à-dire un modèle dont la théorie est l'intersection des théories issues de modèles de graphes. Ils conjecturent que cette théorie est β (le plus petit élément du treillis des λ-théories). Actuellement, ils essayent de prouver que l'intersection de toutes les théories sensibles issues de modèles de graphes est $\mathcal{H}$, la plus petite λ-théorie sensible (les problèmes de trouver des modèles dont la théorie est β (resp. $\mathcal{H}$) est ouvert).

Modèles catégoriques de la logique linéaire. Dans [Mel03c], Melliès propose un article de survol (survey) des multiples définitions catégoriques de modèle de la logique linéaire, où il distingue les définitions basées sur l'aspect comonoïdal (Lafont) ou comonadique (Seely, Barber, Benton, Hyland, de Paiva) de l'exponentielle. Les axiomatiques comonoïdales sont simples mais limitées à certains modèles (où l'exponentielle est une construction libre) tandis que les axiomatiques comonadiques sont générales mais très difficiles à vérifier dans les modèles concrets (de jeux en particulier). Melliès avance donc une nouvelle définition en fin d'article, à la fois générale et commode à vérifier dans les modèles concrets.

Sémantique des jeux et opérateurs de contrôle. Danos et Harmer ont développé une analyse des opérateurs de contrôle en sémantique des jeux, et établi une symétrie inattendue entre les opérateurs de contrôle locaux et globaux [DH01].

Jeux axiomatiques. Dans son mémoire de DEA, Hirschowitz a exploré la possibilité d'étudier un fragment de la théorie des jeux sans pointeurs selon une idée de Ghica et McCusker. La première partie de sa thèse propose une approche catégorique et axiomatique à la sémantique des jeux, cadre qu'il utilisera pour interpréter la logique linéaire.

Sémantique des mondes possibles. Levy a travaillé sur des modèles de mondes possibles pour Call-By-Value (CBV) [Lev02], ainsi que, en collaboration avec Thielecke et Power, sur la sémantique catégorique de CBV [LTP04]. Ce dernier travail soulève des questions inattendues de cohérence, dont il a pu discuter avec Eugenia Cheng durant l'une des visites de cette dernière à PPS.

2 Thème II : Théorie de la démonstration et λ-calcul

Participants : Sylvain Baro, Pierre-Louis Curien, Jean-Baptiste Joinet, Thierry Joly, Olivier Laurent, Paul Blain Levy, Ralph Matthes, François Maurel, Paul-André Melliès, Alexandre Miquel, Raphaël Montelatici, Vincent Padovani, Michel Parigot, Emmanuel Polonovski.

Ce thème est placé sous le signe de la correspondance de Curry-Howard, déjà célébrée dans l'introduction, et de la logique linéaire (LL) (cf. thème I). Rappelons ici que la logique linéaire apparaît comme un raffinement de la logique intuitionniste, puis de la logique classique, en donnant un statut "logique" aux règles structurelles à travers les connecteurs exponentiels ! et ?. Ainsi l'implication linéaire $\multimap$ agit sans duplication ni effacement et il est nécessaire d'introduire la mention explicite de ces connecteurs exponentiels pour retrouver l'implication intuitionniste par $A \to B = !A \multimap B$ ou l'implication classique par $A \to B = !?A \multimap ?B$. Cette décomposition a permis une plus grande précision dans l'analyse du calcul en logique, qui a mené en particulier à une vue plus géométrique et plus interactive des preuves (réseaux de preuve, géométrie de l'interaction, sémantique des jeux, etc.). De plus, avec la négation linéaire, on retrouve dans LL une véritable dualité $A^{\perp\perp} = A$ qui est absente de la logique intuitionniste et permet d'analyser le cas classique $\neg\neg A \simeq A$ en expliquant son contenu calculatoire. C'est cette symétrie vue comme une dualité entre programmes et contextes qui est au centre des interprétations interactives de la logique.

Autour de cette thématique forte de la correspondance de Curry-Howard rendue plus précise et plus riche grâce à la logique linéaire, nous distinguerons plusieurs grandes lignes de recherche. Des travaux convergents ont approfondi la prise en compte du contenu calculatoire du raisonnement classique et des propriétés de dualité qui en découlent (section 2.1). Sur le plan de la logique linéaire proprement dite, l'aspect syntaxique central concerne les réseaux de preuve (section 2.2). À partir de théories construites sur un noyau de λ-calcul, se développent des questions de fondements dont on cherche à mieux comprendre les rapports avec la théorie des ensembles (section 2.3). De nombreux résultats sur le λ-calcul simplement typé résolvant des conjectures des années 80 ont été obtenus (section 2.4). Enfin, des contributions plus variées sont résumées dans la section 2.5.

2.1 Logique classique : dualités et polarisation

L'extension de la correspondance de Curry-Howard à la logique classique à partir des travaux de Griffin (voir aussi thème III) permet de donner une

interprétation algorithmique des preuves utilisant le raisonnement classique, liée aux opérateurs de contrôle des langages fonctionnels.

Plus qu'une simple extension, cette correspondance de Curry-Howard classique est en fait plus riche que dans le cas intuitionniste. Là où l'on reconnaît habituellement *une* notion de calcul associée à la logique intuitionniste à travers le λ-calcul, on découvre *trois* interprétations calculatoires de la logique classique. Du fait de la symétrie entre hypothèses et conclusions donnée par la négation dans les séquents de la logique classique, l'élimination d'une coupure entre une hypothèse et une conclusion peut se faire dans deux directions différentes. On obtient alors trois interprétations selon que l'on choisit de manière systématique l'une ou l'autre de ces deux directions (logique classique en *appel par nom* ou en *appel par valeur*) ou bien que l'on décide d'autoriser à tout moment à choisir l'une ou l'autre de ces directions (logique classique *non déterministe*). Comme témoins historiques de ces trois notions de calcul, on peut citer les $\lambda\mu$-calculs (Parigot) par nom et par valeur et le λ-calcul symétrique (Berardi).

Dualité entre appel par nom et appel par valeur. Selinger [Sel01] a montré que les deux interprétations calculatoires déterministes de la logique classique peuvent être vues comme duales l'une de l'autre. Cette dualité est définie de manière catégorique : les catégories de contrôle (une extension des catégories cartésiennes fermées) constituent les modèles de la logique classique en appel par nom, et leurs duales (catégories opposées), les catégories de co-contrôle, sont les modèles de l'appel par valeur. Elle admet également une présentation syntaxique sous la forme de traductions du $\lambda\mu$-calcul par nom dans le $\lambda\mu$-calcul par valeur, et réciproquement, qui sont inverses l'une de l'autre.

Différents calculs issus d'approches souvent très différentes ont permis de faire apparaître également cette dualité. On en distinguera deux sortes : d'une part des calculs déterministes avec dualité qui, à l'aide d'une unique procédure de calcul, permettent de coder aussi bien l'appel par nom que l'appel par valeur grâce à l'utilisation d'une traduction appropriée (par exemple LLP ou CBPV). D'autre part des calculs non déterministes dans lesquels le choix d'une stratégie de réduction donne accès à l'appel par nom ou à l'appel par valeur (par exemple le $\bar{\lambda}\mu\tilde{\mu}$-calcul ou le $\lambda\mu$-calcul avec réductions μ et $\tilde{\mu}$).

La logique linéaire polarisée (LLP, cf. thème I) est issue de la distinction entre connecteurs réversibles (ou négatifs) et irréversibles (ou positifs), dans la suite des travaux d'Andreoli et Girard. En simplifiant les traductions étudiées par Danos-Joinet-Schellinx [DJS97] de la logique classique dans la logique linéaire, Laurent a défini des traductions directes de la logique classique en appel par nom (traduction négative : $A \to B = !A \multimap B$)

et en appel par valeur (traduction positive : $A \to B = !(A \multimap ?B)$) dans LLP [Lau02a, Lau03c]. La dualité entre les deux modes de calculs correspond ainsi simplement à la dualité donnée par la négation linéaire dans LL et qui permet de passer des connecteurs positifs aux connecteurs négatifs dans LLP [Lau03c]. Inversement, une preuve de LLP peut être relue comme deux $\lambda\mu$-termes selon que l'on met en avant les connecteurs positifs ou négatifs et l'on peut reconstruire ainsi les traductions syntaxiques de Selinger.

Le paradigme Call-By-Push-Value (CBPV) [Lev99, Lev] a émergé d'une analyse attentive des paradigmes d'appel par valeur et d'appel par nom dans les langages fonctionnels : à travers divers modèles utilisant des domaines, des continuations, des jeux, de la mise en mémoire ou des mondes possibles, on retrouve une même décomposition naturelle pour l'interprétation de l'appel par nom et de l'appel par valeur. C'est cette observation qui a conduit Levy à proposer le langage CBPV qui prend pour primitives les constructions plus atomiques ainsi mises à jour. CBPV admet deux espèces de types, appelés respectivement *value types* et *computation types*. Levy a défini des traductions de l'exécution en appel par valeur (basée sur les values types) et en appel par nom (basée sur les computation types) qui permettent de simuler les deux paradigmes dans CBPV. Il a également rapproché la sémantique opérationnelle dérivée de son modèle de jeux pour CBPV avec contrôle du travail de Danos, Herbelin et Regnier sur les sémantiques de jeux et les machines abstraites, ainsi que des arbres de Böhm abstraits de Curien et Herbelin (voir thème I), dont elle est une instance.

À partir d'une analyse du λ-calcul et de son pendant en calcul des séquents, le $\bar{\lambda}$-calcul, Curien et Herbelin [CH00] ont construit une syntaxe universelle non déterministe (le $\bar{\lambda}\mu\tilde{\mu}$-calcul) permettant de simuler les réductions du calcul des séquents classique. Ils donnent un rendu saisissant de la dualité entre appel par valeur et appel par nom à travers les deux orientations possibles d'une paire critique symétrique. Cette approche a été comparée par Rocheteau [Roc02] à celle basée sur la déduction naturelle qui consiste à mélanger $\lambda\mu$-calcul par nom et par valeur en un unique calcul non déterministe.

Une retombée de ces travaux devrait être une meilleure compréhension de ce qu'est un calcul fonctionnel par valeur (syntaxe typée ou non et règles de calcul).

Le foisonnement de calculs où la dualité entre appel par nom et appel par valeur est mise en avant rend nécessaire de comparer ces systèmes, d'autant qu'ils proviennent de lignes de recherche assez diverses. Des premiers pas ont déjà été effectués dans cette direction :

– Laurent a étudié les plongements du $\lambda\mu$-calcul et du $\bar{\lambda}\mu\tilde{\mu}$-calcul dans LLP [Lau02a, Lau03c],

– Curien-Herbelin [CH00] et Rocheteau [Roc02] ont comparé le $\lambda\mu$-calcul et le $\bar{\lambda}\mu\tilde{\mu}$-calcul,
– en établissant un parallèle entre value types (resp. computation types) de CBPV et formules positives (resp. négatives) de LLP, Laurent et Levy ont établi une correspondance entre JWA (Jump-With-Argument, un langage plus simple que CBPV et équivalent à ce dernier en présence d'opérateurs de contrôle [Lev01]) et LLP. Ceci invite à approfondir la compréhension du rôle de la polarisation dans la structure de certains modèles. L'exemple le plus frappant est celui des jeux non parenthésés de Laurent (cf. thème I). Mais d'autres exemples intrigants sont fournis par des modèles de jeux parenthésés [Lev03] et de 'lower powerdomains' qui sont clairement polarisés, mais sont des modèles de CBPV et non de JWA. Ces observations semblent indiquer qu'en toute généralité les distinctions value/computation et positif/négatif sont indépendantes.

Il apparaît ainsi que tous ces calculs sont essentiellement équivalents, mais les rapports exacts restent à étudier.

Traductions CPS et par double négation. L'extension de la correspondance de Curry-Howard à la logique classique met en relation les $\neg\neg$-traductions (qui permettent de plonger la logique classique dans la logique intuitionniste) et les traductions CPS (Continuation-Passing-Style, qui permettent d'encoder les opérateurs de contrôle des langages fonctionnels dans le λ-calcul).

Dans le cadre de l'étude du $\bar{\lambda}\mu\tilde{\mu}$-calcul [CH00], Curien et Herbelin ont défini des CPS-traductions de leur calcul dans le λ-calcul qui permettent de simuler les deux stratégies de réduction par nom et par valeur à l'aide du λ-calcul, donnant une forme de validation des règles de calcul.

Dans [LR03], Laurent et Regnier ont montré que, grâce aux traductions directes de la logique classique dans LLP, il est possible de définir un analogue des $\neg\neg$-traductions à l'intérieur même de la logique linéaire. Il existe deux telles traductions : la traduction par boîtes qui correspond aux CPS-traductions de Plotkin et la traduction par renversement qui correspond à la $\neg$-traduction de Krivine. Il en découle une classification des $\neg\neg$-traductions (de la logique classique dans la logique intuitionniste) en deux familles, selon qu'elles sont envoyées sur l'une ou l'autre des traductions linéaires. On conjecture que toute $\neg\neg$-traduction se trouve dans l'une de ces deux classes, ce qui montrerait que la grande variété de ces traductions ne cache en fait que deux contenus calculatoires distincts. Ce travail devrait permettre d'analyser d'autres mécanismes de contrôle à l'aide de contraintes de linéarité naturelles à définir dans LL.

Terminaison et non déterminisme. Le système $\mathrm{LK}^{\mathrm{tq}}$ a été introduit par Danos, Joinet et Schellinx pour comparer logique classique et logique linéaire. Joinet, Schellinx et Tortora de Falco [JSTdF02] ont donné une preuve de forte normalisation de la tq-réduction pour toutes les versions standard du calcul des séquents pour les logiques classique et intuitionnistes (premier et second ordre), et un argument concis montrant la complétude de la contrainte de focalisation pour les preuves en calcul des séquents. Cette contrainte de focalisation est liée aux travaux sur la polarisation et permet de définir le sous-système $\mathrm{LK}^{\eta}_{\mathrm{pol}}$ de $\mathrm{LK}^{\mathrm{tq}}$ qui reste complet pour la logique classique, mais dont le plongement dans LL nécessite l'introduction de coupures. Laurent, Quatrini et Tortora de Falco [LQTdF00] ont montré qu'en utilisant LLP plutôt que LL il n'est plus nécessaire de recourir à ces coupures et qu'on en déduit une preuve de terminaison de la réduction de $\mathrm{LK}^{\eta}_{\mathrm{pol}}$ à l'aide des réseaux de preuve polarisés.

Dans son travail de thèse, Polonovski prouve la terminaison du $\overline{\lambda}\mu\tilde{\mu}$-calcul avec substitutions explicites. Il s'agit de la première preuve de terminaison d'un calcul non-déterministe avec substitutions explicites. La technique de preuve utilise entre autre une technique appelée "PSN implique SN", initialement suggérée par Herbelin. Cette technique simple et qui peut être utile dans d'autres calculs consiste à établir la propriété de forte normalisation souhaitée pour le calcul avec substitutions explicites comme conséquence de deux propriétés :

- la forte normalisation du calcul sans substitutions explicites (prouvée par une variante de la méthode de Barbanera et Berardi, voir ci-dessous) ;
- la propriété de préservation de la normalisation forte (PSN) qui dit que si un terme sans substitutions explicites normalise fortement dans le calcul sans substitutions explicites, alors il normalise fortement dans le calcul avec substitutions explicites.

Parigot a lui aussi étudié des interprétations calculatoires non confluentes de la logique classique. Dans [Par00a], il étudie la possibilité de donner une interprétation calculatoire de la logique classique avec une négation involutive. Il montre qu'il n'est pas possible de le faire en ajoutant $\neg\neg A = A$ au λ-calcul typé, mais qu'en revanche cela est possible avec le $\lambda\mu$-calcul. Le résultat est le $\lambda\mu$-calcul symétrique, qui entretient des liens étroits avec le λ-calcul symétrique introduit par Berardi. Dans [Par00b], Parigot montre la normalisation forte du λ-calcul symétrique du second ordre. La preuve est une généralisation élégante commune des preuves de normalisation forte du $\lambda\mu$-calcul du second ordre et du λ-calcul symétrique du premier ordre de Berardi. Il s'agit de la première preuve de normalisation d'un système non confluent du second ordre.

2.2 Réseaux de preuves.

Principale nouveauté syntaxique liée à la logique linéaire, les réseaux de preuves permettent une représentation plus canonique des démonstrations formelles : des preuves du calcul des séquents qui ne se distinguent que par des commutations de règles "anodines" sont identifiées. Les réseaux sont basés sur une représentation des preuves par des graphes, et non plus par des arbres de dérivation comme c'est le cas en calcul des séquents ou en déduction naturelle. Le prix à payer est que tous ces graphes de preuves (appelés *structures de preuves*) ne sont pas valides. Alors qu'il est immédiat de vérifier qu'une preuve en calcul des séquents est correctement formée (par une procédure purement locale qui vérifie la bonne application des règles), il est nécessaire de vérifier une condition globale sur les structures de preuves : un *critère de correction*, pour s'assurer qu'il s'agit bien d'une démonstration. Cette représentation graphique a permis de révéler certaines propriétés "géométriques" des preuves, invisibles dans les présentations syntaxiques par des arbres de dérivation.

On peut résumer en trois grandes questions l'étude des réseaux de preuves :

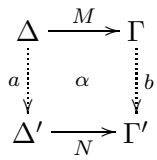
- trouver la bonne représentation graphique des preuves (qui identifie le plus de preuves sans devenir incohérente) ;
- définir des critères de correction pour cette syntaxe qui, par des propriétés de graphes, caractérisent les structures de preuves qui correspondent à des dérivations du calcul des séquents ;
- étudier la procédure d'élimination des coupures à l'intérieur de cette syntaxe de réseaux où la suppression d'une part de la séquentialité des preuves simplifie grandement le travail.

Le seul cas où ces trois questions ont trouvé des réponses optimales (et ce dès l'introduction des réseaux de preuves) est le fragment multiplicatif de la logique linéaire (MLL) sans constantes. On distingue en particulier deux principaux critères de correction : le critère du *long chemin* de Girard et le critère *acyclique et connexe* de Danos-Regnier. La prise en compte des connecteurs exponentiels (MELL) et des constantes multiplicatives est également considérée comme satisfaisante et n'a pas donné lieu à des développements récents. Les deux directions essentielles de travail restent la gestion des connecteurs additifs (aussi bien en ce qui concerne la question de la représentation que celles du critère de correction et de l'élimination des coupures) et d'autre part la logique non commutative multiplicative (MNL) essentiellement dans le domaine des critères de correction.

Catégories doubles : une description modulaire des réseaux MLL. En logique linéaire multiplicative, on appelle *module* une région dans un

réseau de preuve. La frontière qui sépare un module de son environnement est appelée *interface*. A travers ces interfaces, se déroulent des "interactions" entre modules, liées à l'élimination des coupures.

Pour décrire cette situation, Melliès a introduit une catégorie double [Mel02d], c'est-à-dire une "catégorie" dont les cellules ont une forme en carré :



où on écrit $\alpha : a \Rightarrow_h b$ ou $\alpha : M \Rightarrow_v N$

Les cellules peuvent être composées horizontalement ou verticalement, de même que les morphismes horizontaux, et verticaux. Dans cette catégorie double définie pour MLL,

- les objets Δ, Γ sont les *interfaces*,
- les morphismes horizontaux $M : \Delta \longrightarrow_h \Gamma$ sont les *modules*,
- les morphismes verticaux $a : \Delta \longrightarrow_v \Delta'$ sont les *interactions* à travers l'interface Δ, dans l'esprit de la SOS (Structured Operational Semantics) de Plotkin,
- les cellules sont les étapes de réécriture, ou d'interaction à travers l'interface.

Cette structure double catégorique permet de dégager, dans un cas simple – celui de la logique linéaire multiplicative –, les points de contact entre réécriture et sémantique par interaction (cf. sémantique des jeux, thème I). La même structure a été étudiée par Gadducci et Montanari d'une part, et plus récemment par Milner, dans le cadre différent, mais proche, des calculs de processus.

Réseaux polarisés. La définition d'une syntaxe de réseaux pour le fragment polarisé de la logique linéaire est obtenue de manière immédiate en restreignant les réseaux de MELL aux formules polarisées (MELLP). Laurent a montré [Lau03c] que, dans ce contexte, il est possible de simplifier le critère de correction de Danos-Regnier en introduisant une orientation des arêtes. De plus, ce critère permet de prendre en compte les constantes multiplicatives.

Pour les connecteurs additifs, différentes présentations et techniques ont été explorées : poids additifs [Lau99], boîtes additives [LQTdF00] et tranches additives [LTdF01]. Dans chaque cas, on a pu définir une procédure d'élimination des coupures, grâce à contrainte de polarisation, alors que ceci n'est pas possible dans le cas général de LL.

La notion de tranche a été introduite par Girard pour les réseaux de LL sans qu'un critère de correction satisfaisant ne puisse être défini excepté très récemment pour MALL par Hughes et van Glabbeek. Dans le cas polarisé, Laurent et Tortora de Falco ont donné un critère de correction pour des réseaux à tranches en présence de tous les connecteurs linéaires et ont montré que l'on obtient une représentation canonique des preuves au sens où l'interprétation dans la sémantique cohérente (pour le fragment de LL correspondant au λ-calcul avec paires) [LTdF01] et dans la sémantique des jeux (pour tout le fragment polarisé de LL) [Lau03d] est injective pour les réseaux sans coupure.

Grâce à l'orientation sous-jacente à la notion de réseaux polarisés, Montelatici a montré qu'en retirant la condition d'acyclicité du critère de correction (rendant presque toute structure de preuve correcte), on obtient un système de logique polarisée avec cycles qui correspond à la sémantique des catégories à traces et permet d'ajouter des points fixes à LLP.

Critère topologique pour les réseaux non-commutatifs. Melliès a proposé dans [Mel02e] une présentation en "ruban" des démonstrations (proche de celle en "CW-complexes" définie par Burroni et Métayer) dans laquelle les critères de Girard et Danos-Regnier découlent d'un seul critère topologique, qui s'énonce de la sorte : tout choix de basculement (switching) des connecteurs 'tenseur' et 'par' définit une surface S homéomorphe au disque. Il apparaît alors que cette surface S a pour bord unique le long chemin (long trip) défini dans le critère de Girard, et est contractile comme l'arbre construit par le critère de Danos et Regnier.

Cette reformulation topologique s'applique aussi au critère de correction introduit par Abrusci et Ruet pour la logique non commutative MNL. Rappelons que la logique MNL est une extension conservative de MLL et de la logique cyclique multiplicative McyLL, définie en restreignant la règle de commutation de MLL aux permutations *cycliques*. Un critère bien connu caractérise les démonstrations de McyLL comme les démonstrations *planaires* de MLL, dont les formules de conclusion sont placées sur le bord extérieur de la démonstration. De même, Melliès montre [Mel02e] qu'une fois reformulé topologiquement, le critère d'Abrusci et Ruet caractérise les démonstrations de MNL comme les démonstrations de MLL pour lesquelles tout choix de basculement des connecteurs multiplicatifs définit une surface *planaire*, dont les formules de conclusion sont placées sur le bord extérieur. Une démonstration purement topologique de l'élimination des coupures dans la logique MNL en découle.

Les réseaux de preuves apparaissent également comme un outil clef dans différents domaines, comme l'étude du λ-calcul avec substitutions explicites

(voir thème IV), l'analyse des systèmes logiques à complexité bornée (voir section 2.5), etc. .

2.3 Théories fondationnelles

Antifondation et Calcul des Constructions avec univers. Dans la troisième partie de sa thèse [Miq01], Miquel montre que le système de types purs $F\omega.3$ (un fragment du Calcul des Constructions avec univers) contient la théorie des ensembles de Zermelo intuitionniste (IZ), au moyen d'une représentation inspirée de la théorie des hyper-ensembles d'Aczel (la théorie des ensembles basée sur l'axiome d'antifondation). Ce résultat, qui résout en particulier une conjecture de Coquand, donne naturellement un contenu calculatoire fortement normalisable aux preuves de IZ par traduction. Dans [Miq03], Miquel étend ce résultat à la théorie des ensembles de Zermelo-Fraenkel intuitionniste (IZF) au moyen d'une interprétation calculatoire d'une forme intuitionniste du symbole de choix de Hilbert, laquelle permet d'obtenir une réalisation de l'axiome de collection intuitionniste et un théorème de normalisation forte pour IZF. Depuis, Miquel étudie le modèle de réalisabilité sous-jacent pour le comparer au modèle de réalisabilité classique de ZF de Krivine (voir thème III).

Dans [MW03], Miquel et Werner montrent que la preuve de correction du modèle booléen standard du Calcul des Constructions (réputé pour sa simplicité) est erronée en raison d'une difficulté technique insoupçonnée liée à l'interaction du polymorphisme et des types dépendants, et proposent une méthode pour la corriger.

Map Theory. La Map Theory (MT) de Klaus Grue est un système fondationnel conçu comme une extension du λ -calcul non typé, sur lequel beaucoup de questions intéressantes sont ouvertes. La puissance de MT est au moins celle de ZFC+$BonneFondation$. Vallée a établi une version de MT destinée à jouer pour l'Antifondation le rôle que la MT de Grue joue pour la fondation et qui permet donc de définir des types coinductifs et de faire du raisonnement par coinduction [Val01], et il en a construit un modèle [Val03].

Le travail sur MT s'est aussi développé sur un plan plus syntaxique :

– Dans son article fondateur [Gru92], Grue a produit une interprétation syntaxique des axiomes de ZFC (plus l'axiome de bonne fondation) comme termes de MT, et une interprétation du calcul des prédicats du premier ordre dans le raisonnement équationnel de MT qui permet d'y valider les axiomes de ZFC. Vallée a adapté cette traduction (en la reprenant et la clarifiant) au cadre antifondationnel, et a pu de plus apporter des réponses à la question de savoir quelles théories du premier ordre sont

susceptibles d'être interprétées et validées (directement) par MT ou des fragments naturels de MT.

– Vallée s'est par ailleurs attaché à répondre à une problématique de Furio Honsell sur la possibilité de représenter le système fondationnel de De Giorgi dans une extension (à découvrir) de la Map Theory. Le système de De Giorgi est un système extrêmement général qui se veut apte à intégrer des sous-théories provenant de différentes disciplines scientifiques. Il comprend en particulier une théorie des collections et des corrélations qui englobe la théorie des classes de Bernays-Gödel. Contrairement à MT, dont le langage (surtout) et l'axiomatisation sont minimalistes, le système de De Giorgi offre dès l'abord à l'utilisateur d'intégrer tous les concepts dont il peut avoir besoin, et ce à plusieurs niveaux simultanément. Vallée a pu modéliser une partie significative de cette théorie, en utilisant des modèles du lambda-calcul non typé et l'interprétation de l'appartenance caractéristique de MT.

Voici quelques problèmes ouverts en cours d'investigation :

– Berline travaille avec Grue sur la cohérence de sa nouvelle version de MT (également bien fondée), dans laquelle l'univers des "ensembles" est défini (comme point fixe) par un λ-terme, et où n'apparaissent plus les axiomes qui exprimaient la clôture de cet univers par des opérations simples (et naturelles pour le lambda-calcul).
– Evaluation de la force de la version originale de MT par rapport à ZFC (strictement plus forte ? de combien ?). Pour l'instant, on peut seulement en montrer la cohérence en utilisant en plus l'existence d'un cardinal inaccessible [BG97].
– Berline et Vallée ont conçu une version antifondée de MT à la fois plus élégante et plus duale de la MT originelle que celle présentée dans la thèse de Vallée, mais ils n'ont pas encore pu en montrer la cohérence.

Notons que l'excellente expressivité de la Map Theory et les commodités calculatoires qu'elle contient donnent lieu à des applications en informatique qui sont actuellement développées par Grue (et ses étudiants). Exemples : sémantique des langages objets et implantation d'un vérificateur de preuves basé sur MT.

2.4 Lambda-calcul simplement typé

On aurait pu croire la théorie du λ-calcul simplement typé (le plus simple des sytèmes de types, correspondant à la logique intuitionniste minimale) entièrement balisée. Il n'en est rien, et les résultats qui suivent ont reçu toute l'attention des figures "historiques" du domaine (Barendregt, Statman).

Rétractions définissables. Dans [Pad01] Padovani démontre la décidabilité du problème de l'existence d'une rétraction définissable entre deux types simples, dans le cas d'un seul type de base, résolvant ainsi partiellement un problème ouvert depuis 1985. Dans [BL85], Bruce et Longo fournissent une première caractérisation des rétractions définissables à partir de la seule règle de β-réduction, sans la η-réduction, en construisant un système de types engendrant tous les couples de types simples entre lesquels de telles rétractions existent. Le même article présente une extension de ce système dont est démontrée la stricte incomplétude, relativement au problème étudié. Dans [dPS92], de'Liguoro, Piperno et Statman montrent comment étendre ce dernier système pour engendrer tous les couples de types simples entre lesquels existe une rétraction définissable par un λ-terme linéaire. Padovani attaque le problème des rétractions sous un autre angle : au lieu de fournir une nouvelle extension du système précédent, il se sert de l'algorithme de décidabilité du modèle minimal du λ-calcul simplement typé, présenté dans [Pad96], pour construire un nouvel algorithme déterminant, étant donnés deux types simples construits sur un unique type de base, s'il existe une rétraction définissable entre ces types. Le problème des rétractions entre types simples construits avec au moins deux types de base est à ce jour encore ouvert.

Types finiment engendrés. Lors de son séjour post-doctoral à Nijmegen, Joly a poursuivi son étude des fonctions obliquement représentables dans le λ-calcul simplement typé, étude entreprise à l'occasion de sa thèse [Jol00a] et qui lui avait permis de caractériser les fonctions calculables en temps constant parallèle au sein du λ-calcul non typé [Jol01a]. Rappelons qu'une représentation oblique d'une fonction $f : A \to B$ est un λ-terme R de type $A[o := T] \to B$ tel que : $R(M[o := T]) =_{\beta\eta} f(M)$ pour tout λ-terme clos M de type A. En s'attachant plus particulièrement aux surjections obliques, Joly a mis en évidence un préordre induit, qui est assez curieusement semblable au préordre des injections étudié par Statman et Dekkers. De la même façon que ce dernier permet de réduire des problèmes de séparation tels que la complétude du calcul pour une classe de modèles donnée, le préordre des surjections obliques s'avère être un outil précieux pour plusieurs problèmes fondamentaux d'existence, tels que la définissabilité. Cela a permis à Joly d'établir que certains types ne sont engendrés par aucune base combinatoire finie [Jol00b] ; cette question avait été soulevée en 1985 par Statman. Par la suite, Joly a obtenu diverses caractérisations des types finiment engendrés et un algorithme décidant cette propriété [Jol01b, Jol03a]. En transposant sa notion de représentation oblique aux modèles pleins finis, il a aussi établi l'indécidabilité du dernier cas ouvert du problème de la définissabilité, celui du modèle construit à partir de deux atomes [Jol03b].

2.5 Varia

Temps NP et logique. Maurel a trouvé une caractérisation logique de la complexité NP à l'aide d'une extension de la logique légère affine ndILAL [Mau03a] représentée à l'aide de réseaux de preuves. Les machines de Turing polynomiales non déterministes sont plongeables dans ndILAL et, en un certain sens, tout réseau de preuve de ndILAL est réductible non déterministiquement en temps polynomial. La caractérisation est similaire aux caractérisations du temps polynomial par les logiques légères (Girard). Il s'agit, semble-t-il, de la première caractérisation "à la Curry-Howard" de la complexité NP. Une propriété d'équivalence sur les programmes en ndILAL est aisément montrée équivalente au célèbre problème P = ? NP.

$\lambda = q + \nu$. Baro et Maurel ont conçu un calcul mélangeant simplicité des constructions et richesse d'expression. Leur syntaxe (le $q\nu$-calcul) est basée sur un λ-calcul où le λ est séparé en deux : un lieur ν permettant la déclaration d'une variable locale, et un combinateur q admettant une réduction semblable à celle du λ-calcul, mais où les variables sont capturées.

Ce calcul offre une meilleure compréhension de la capture de variables, souvent considérée comme un phénomène "parasite". Il permet de coder certaines constructions usuelles des langages de programmation (par exemple, les exceptions de ML). Il ouvre la possibilité :

- d'établir des rapports entre ces constructions et d'en factoriser l'analyse (on peut citer les exceptions, la portée dynamique, les continuations, les variables fonctionnelles croissantes...) ;
- de formaliser de nouveaux opérateurs de contrôle pour les langages de programmation.

Ces travaux sont en cours, et font l'objet d'une prépublication [BM03b].

A propos de la règle ξ. Répondant à une question soulevée par Wadler en 2001, Joly a établi que la règle (ξ) du λ-calcul non typé (qui énonce la λ-congruence du calcul) n'est pas finiment axiomatisable [Jol03c]. Ce résultat repose sur le fait que l'absence de (ξ) est équivalente au remplacement de la règle (β) par une variante faible. Cette dernière variante possède une notion de réduction viable et donne naissance à un calcul plus proche de la logique combinatoire que du λ-calcul proprement dit.

Types de données d'ordre supérieur imbriqués. Matthes a poursuivi lors de son séjour postdoctoral à Paris un travail sur les types de données d'ordre supérieur imbriqués. Il utilise le cadre du système $F\omega$ de Girard (po-

lymorphisme paramétrique d'ordre supérieur) pour donner une formulation générale, mais garantissant la terminaison, de l'itération (resp. co-itération) et de la récursion (resp. co-récursion) primitive sur des plus petits (resp. plus grands) points fixes de constructeurs de type de niveau arbitrairement élevé.

Aspects philosophiques de la théorie de la démonstration. Joinet a développé les thèses suivantes dans les articles [Joi02, Joi03].

- Le "tournant mathématique" de la fin du XIXème siècle avait transformé la logique en une théorie de la vérité et des preuves, loin de sa définition philosophique comme "théorie du raisonnement". Avec le remplacement récent du paradigme traditionnel des *preuves-comme-discours* par celui des *preuves-comme-programmes*, la logique s'est transformée en une "théorie des fondements du calcul". On pourrait interpréter cette seconde métamorphose comme le signe d'une "dérive" supplémentaire de la logique vers la technologie, toujours plus proche de l'ingénierie pratique, toujours plus loin du raisonnement comme processus intellectuel. Joinet évalue le point de vue inverse, qui replace le raisonnement en temps que processus dynamique au coeur de la logique.
- Au sens de la tradition logique prédominante aux XIXème et XXème siècles, *temps* et *logique* sont sans interaction. Les racines algébristes de la logique moderne et le rayonnement de la définition tarskienne de la notion de "tautologie" enfermèrent en effet durablement la logique dans son statut de théorie de lois formelles immuables, évacuant toute forme de temporalité. La place toujours plus centrale occupée dans les recherches contemporaines sur les preuves par la dynamique de l'évaluation (dynamique de l'heuristique, dynamique de la conversion des preuves) tend au contraire à installer le temps au cœur même du logique. Analysant ce renversement, Joinet propose d'aller jusqu'à redéfinir la logique comme la domestication du temps rationnel.

3 Thème III : Spécifications et réalisabilité

Participants : Emmanuel Beffara, Vincent Danos, Jean-Louis Krivine, Olivier Laurent, Paul-André Melliès, Jérôme Vouillon.

L'idée directrice de ce thème est de prendre au pied de la lettre la correspondance de Curry-Howard et de résoudre le *problème de la spécification*, qui se pose pour chaque théorème de mathématiques, et qui consiste à trouver la spécification qui lui est associée. C'est, bien entendu, un projet très ambitieux et de longue haleine. Par exemple, il y a seulement dix ans, on ne connaissait la solution que dans des cas assez triviaux, essentiellement les

théorèmes d'arithmétique Π^0_2, et ce, dans un cadre purement intuitionniste. Le passage indispensable à la logique classique (aucun mathématicien n'utilise la logique intuitionniste) a été rendu possible par la découverte, par T. Griffin en 1990, du typage des instructions de contrôle de LISP par le tiers exclu.

Grâce aux travaux de Krivine, il a été possible de construire peu à peu un cadre logique dans lequel on peut poser, de façon correcte et intuitive, le problème de la spécification en logique classique. Le point de départ a été une idée simple de machine pour l'exécution du λ-calcul [Kri03b] conçue en 1983 et qui a acquis depuis une certaine notoriété (un numéro de la revue "Higher Order and Symbolic Computation" va lui être prochainement consacré [Dan03]). Cette machine joue un rôle central dans la théorie, car c'est à la fois un objet mathématique qui reflète la structure des démonstrations, et un objet informatique où l'on retrouve les notions habituelles des langages de programmation impératifs. Elle permet d'interpréter toutes les preuves en logique intuitionniste et, avec l'adjonction d'instructions de contrôle, celles en logique classique.

Mais les preuves mathématiques ne se font pas en logique pure. L'étape suivante consiste donc à associer à chaque axiome de la théorie des ensembles un programme ou une nouvelle instruction pour cette machine. Cela a été fait dans [Kri01] pour les axiomes de Zermelo-Frænkel, *sauf l'axiome du choix*. La difficulté essentielle provient de l'axiome d'extensionnalité. On formalise donc d'abord une théorie des ensembles sans extensionnalité, dans laquelle on peut définir, soit par induction sur la classe des ordinaux, soit par coinduction, c'est-à-dire par bisimulation, une nouvelle relation d'appartenance qui est extensionnelle. On parvient ainsi à réaliser les axiomes de ZF par des programmes d'ailleurs fort complexes, mais il est remarquable qu'aucune nouvelle instruction ne soit nécessaire.

Reste à traiter l'axiome du choix. Sa forme de loin la plus utilisée est l'axiome du choix *dépendant*, qui suffit pour formaliser l'analyse classique (fonctions de variable réelle ou complexe, transformation de Fourier, probabilités, etc). Une solution assez surprenante a été trouvée tout récemment, dans [Kri03c] : elle consiste à introduire une nouvelle instruction, qui n'est autre que l'*horloge* de la machine, et à typer cette instruction à l'aide d'une proposition qui implique le choix dépendant *en logique classique*. Le problème reste ouvert pour l'axiome du choix général, mais les résultats déjà obtenus ouvrent des perspectives prometteuses.

Mais même des théorèmes très simples correspondent à des spécifications intéressantes : dans [DK00], on montre que les tautologies disjonctives sont associées à des propriétés de communication entre programmes parallèles. Ce travail a été approfondi par Beffara et Danos qui ont identifié une correspon-

dance entre les formes normales disjonctives et une famille de mécanismes de traitement d'exceptions locales [BD03]. Le travail consiste maintenant à généraliser à d'autres calculs les méthodes de réalisabilité employées ici, et à en tirer des méthodes d'étude sémantique pour ces calculs.

Voici quelques *problèmes ouverts* qui sont autant de perspectives :

- Trouver, si elle existe, l'instruction machine associée à l'axiome du choix et sinon, comprendre quelle est la contre-partie informatique de cet axiome (qui peut être bien plus compliquée qu'une simple instruction).
- Comprendre la structure des modèles de ZF qu'on obtient par la méthode de réalisabilité classique. Celle-ci se présente comme une extension tout à fait non triviale de la méthode de forcing de Cohen, où l'ensemble de conditions est une algèbre combinatoire (forcing "non commutatif"). La structure de ces modèles est conditionnée par le paradigme de programmation utilisé (séquentiel ou parallèle, nouvelles instructions, etc).
- Attaquer le problème de la spécification pour des théorèmes convenablement choisis, comme le second théorème d'incomplétude de Gödel, pour lequel le travail est déjà bien entamé.

Ces questions ne présentent pas qu'un intérêt théorique. Une preuve mathématique formalisée et le programme qui lui est associé sont, en effet, des objets syntaxiques de très grande taille et d'une extrême complexité, tout-à-fait comparables aux grands programmes développés dans l'industrie. Mais les mathématiciens ont élaboré un savoir-faire remarquable pour manipuler ces objets de façon à la fois *sûre* et *intuitive*. Il est donc très important de comprendre comment ils font et de *traduire cette expertise dans des programmes*. Bien sûr, l'intérêt philosophique de ce problème n'est pas moins grand : Krivine a développé ce point de vue dans plusieurs articles et conférences (voir par exemple Science & Vie, février 2002).

Types récursifs, sous-typage et réalisabilité. Il est notoirement difficile de définir un langage fonctionnel avec types produits, types fonctionnels, types récursifs, sous-typage, intersection et union de types. On peut espérer que la réalisabilité nous donnera un élément de réponse à ce sujet. Car la réalisabilité renverse l'idée habituelle que les formules précèdent leurs démonstrations : les termes t et leurs environnements π sont introduits en premier, avec une notion d'orthogonalité qui indique que l'interaction entre t et π se déroulera bien. Une valeur de vérité (ou type) est ensuite définie comme un ensemble de termes clos par bi-orthogonalité. On dit alors qu'un terme t *réalise* un type A lorsqu'il en est élément.

Ce renversement permet de partir d'un langage non typé, et de constituer un langage typé, où le sous-typage coïncide avec l'inclusion ensembliste de type.

Qu'en est-il des types récursifs ? Freyd et Pitts ont montré, parmi d'autres, les avantages de considérer un langage de types où plus petit et plus grand point fixe coïncident. Cette coïncidence n'est pas vérifiée dans les modèles de réalisabilité en général : typiquement, le type *Liste d'entiers* et le type *Flot d'entiers* sont distingués parce qu'un certain environnement π termine sur toutes les listes d'entiers, tandis qu'il boucle sur tout flot infini. Pour obtenir la coïncidence, il faut considérer une notion d'orthogonalité fondée sur le test d'erreur (cf. thème I) et non pas sur le test de terminaison.

Dans le cadre de son travail sur XDuce, Vouillon s'est joint à Melliès pour étudier deux familles de modèles de réalisabilité avec types récursifs. La première s'appuie sur des techniques traditionnelles de stratification [MPS86] nécessitant certains opérateurs de test dans le langage ("typecase"). La seconde évite ces opérateurs de test, mais nécessite des techniques plus fines de construction des types récursifs. Ces travaux seront détaillés dans [VM04, MV03]. Un problème intéressant qui reste à mener est l'étude des relations logiques à la Plotkin et Pitts dans ce cadre de réalisabilité.

Rapports avec la sémantique dénotationnelle. On peut aussi vouloir remplacer les λ-termes de la réalisabilité à la Krivine par des objets moins syntaxiques, et plus algébriques, provenant de la sémantique dénotationnelle :

- Chroboczek a développé dans cet esprit un modèle du sous-typage et des types récursifs, fondé sur une sémantique des jeux *non typé* et d'un test d'erreur (cf. thème I). La nature exacte du lien entre le modèle de jeux qu'il obtient et les modèles de réalisabilité habituels reste néanmoins à éclaircir.
- L'extension à la logique classique de la sémantique dénotationnelle et de la correspondance de Curry-Howard n'a été menée jusqu'ici que dans un cadre typé, avec la notion de modèles catégoriques de la logique classique (catégories de contrôle). Le cas non typé semble avoir été ignoré jusqu'ici. Laurent a donné un système de types intersections pour le $\lambda\mu$-calcul dont découle une version classique du modèle de Engeler pour le $\lambda\mu$-calcul pur. De tels modèles dénotationnels du $\lambda\mu$-calcul pur devraient permettre de s'abstraire de certaines considérations syntaxiques dans la théorie du forcing non commutatif de Krivine.

Les relations entre cette problématique de la réalisabilité et d'autres recherches menées dans le laboratoire sont donc nombreuses (voir aussi les travaux de Miquel, section 2.3 , de Maurel, section 1.4, et de Danos, section 6.3).

4 Thème IV : Réécriture et géométrie du calcul

Participants : Vincent Balat, Albert Burroni, Pierre-Louis Curien, Roberto Di Cosmo, Thomas Dufour, Julien Forest, Delia Kesner, Stéphane Lengrand, Paul-André Melliès, Emmanuel Polonovski.

La *théorie de la réécriture* s'intéresse à la transformation discrète, c'est-à-dire pas-à-pas, d'objets de nature calculatoire. Elle embrasse une grande variété de formalismes et de structures : les mots, les termes du premier ordre, les termes d'ordre supérieur (c'est-à-dire avec variables liées), les graphes (par exemple, les réseaux de Petri), ou certains objets géométriques discrets plus complexes à la base d'une théorie naissante de la réécriture n-dimensionnelle. Le premier système de réécriture à avoir été étudié en tant que tel est probablement le λ-calcul, et la théorie générale lui a emprunté des notions clef comme celle de *forme normale*, représentant un objet qui ne peut plus être transformé, et devient donc un candidat tout désigné pour représenter la *valeur* d'un objet, ou encore celle de *confluence*, qui permet de garantir que cette valeur est unique.

La réécriture est utilisée aujourd'hui dans une grande variété de contextes : modélisation des langages fonctionnels ou objets, construction de procédures efficaces de décision pour des théories équationnelles, transformations de programmes, assistants de preuves, etc. . On peut, par exemple, comme l'ont fait Hardin, Maranget et Pagano [HMP98], démontrer la correction de machines abstraites à l'aide du $\lambda\sigma$-calcul (un λ-calcul enrichi d'une opération explicite de substitution issu des travaux de Curien sur les modèles catégoriques du λ-calcul , cf. thème I). L'étude syntaxique du $\lambda\sigma$-calcul (ou de ses variantes) a fait l'objet d'une attention particulière dans le laboratoire (et ailleurs), et a permis, notamment, de le rapprocher des réseaux de preuves (cf. thème II).

On peut aussi aller plus loin et rechercher une théorie uniforme de la réécriture derrière les théories régionales et très syntaxiques dont nous disposons aujourd'hui. Les systèmes de réécriture actuels ont tendance a être extrêmement compliqués : par exemple le $\lambda\sigma$-calcul a 11 règles de réduction et 11 paires critiques. L'objectif est de remplacer l'approche syntaxique par une approche plus abstraite, et d'envisager la réécriture au "bon niveau de généralité". Cette méthodologie sous-tend plusieurs travaux dans le laboratoire :

– ceux de Kesner en direction de l'unification des calculs de substitutions explicites ou de l'axiomatisation du passage de la réécriture d'ordre supérieur à celle du premier ordre ;
– ceux exposés par Melliès dans une série d'articles sur une théorie diagrammatique des résidus, de la standardisation, des réductions de tête,

etc., affranchie des contraintes syntaxiques.

Ces travaux impriment une direction naturelle au sujet : au-delà des diagrammes se profile une véritable *géométrie du calcul*, fondée sur des principes de cohérence n-dimensionnelle. Le laboratoire dispose à cet égard d'une formidable boîte à outils développée depuis plus de trente ans par Burroni, et de l'expertise de Gaucher (qui vient de nous rejoindre) en topologie algébrique.

Nous rendrons successivement compte des recherches menées sur les substitutions explicites et la réécriture d'ordre supérieur (section 4.1), sur les isomorphismes de types, dont l'étude met souvent en jeu de la réécriture et a amené récemment à des applications intéressantes aux techniques d'évaluation partielle et à l'optimisation de programmes (section 4.2), sur la réécriture axiomatique (section 4.3), et enfin sur la réécriture n-dimensionnelle (section 4.4).

4.1 Substitutions explicites et réécriture d'ordre supérieur

Noyau des langages de programmation fonctionnels, le λ-calcul permet de modéliser le travail d'évaluation fait par les machines abstraites de ces langages. Malheureusement, l'unique règle de réduction

$$(\lambda x.M)N \quad \rightarrow_\beta \quad M[x := N]$$

fait intervenir une notion de substitution globale et instantanée de toutes les occurrences de la variable x dans M par N qui est très éloignée de la réalité des machines abstraites. Pour combler cette distance, Curien et al. introduisirent en 1990, avec le système $\lambda\sigma$, la notion de λ-calcul avec substitutions explicites [ACCL91, CHL96], où la substitution n'est plus une notion instantanée et méta-linguistique, mais au contraire une opération explicite du langage qui s'évalue à l'aide d'une batterie de règles, bien plus proches de ce qui se passe dans les machines abstraites réellement implantées, et donc bien mieux adaptées aux preuves de correction. Ainsi, l'introduction de la notion de substitution explicite a ouvert une nouvelle voie de recherche, parce qu'il était nécessaire de définir des calculs spécialement adaptés aux différentes exigences liées à la vérification des machines abstraites, des logiciels d'aide à la démonstration, etc. Cela demandait des propriétés de confluence, de normalisation ou de préservation de normalisation qui n'étaient pas évidentes à établir : en particulier, Melliès a démontré en 1995 que le $\lambda\sigma$-calcul ne préserve pas la normalisation forte du λ-calcul. Le coupable était l'opération de composition des substitutions, qui peut cacher des accumulations de calculs inutiles. Le premier calcul qui rassemblait toutes ces propriétés a été proposé par David et Guillaume en 1999 [DG01] : ce calcul, appelé λ_l, rend explicite les affaiblissements en plus des substitutions explicites.

Substitutions explicites et réseaux de preuves. Di Cosmo, Kesner et Polonovski, à la suite d'un premier travail dans cette direction [DCK96b], ont fourni une traduction du calcul de David et Guillaume dans les réseaux de preuves de la logique linéaire (cf. thème II) qui a permis de prouver la normalisation forte du λ_l-calcul typé [DCKP03]. Polonovski étudie la possibilité d'adapter des techniques de reductibilité pour prouver directement la normalisation forte de ce calcul.

Cependant, dans le calcul présenté dans [DCKP03], la contraction n'est pas "explicite", car la gestion du partage est propre aux règles additives de la logique intuitionniste sous-jacente. Il était donc naturel de vouloir rendre visible et explicite la contraction afin de rapprocher encore plus le modèle calculatoire du modèle logique. C'est ce que vient de faire Lengrand dans son stage de DEA, où il a défini un calcul avec substitutions, affaiblissements et contraction explicites.

Vers une axiomatisation des substitutions explicites. Kesner s'intéresse à unifier, par le biais d'une approche abstraite et axiomatique, les nombreux formalismes avec substitutions explicites existants. Des propriétés des calculs avec substitutions explicites suffisantes pour donner une sémantique correcte à l'opération de substitution ont été identifiées. Ceci est un premier pas vers l'unification de plusieurs formalismes, comme σ [ACCL91], $\sigma_{\Uparrow}$ [HL89], s [KR95], se [KR97], υ [BBLRD96], d [Kes96], ϕ [Muñ97], qui semblaient parfois diverger. Kesner a étudié deux notions de réduction (β et $\beta \cup \eta_{exp}$) sur les termes sans méta-variables dans les systèmes avec substitutions explicites. La confluence de ces deux notions de réduction est démontrée grâce à l'axiomatisation des substitutions explicites [Kes00]. Il reste à étendre cette étude à d'autres propriétés telles que la normalisation forte des termes typés, la préservation de la normalisation, la standardisation.

Réécritures d'ordre supérieur et du premier ordre. Les systèmes de réécriture d'ordre supérieur permettent de décrire des opérateurs avec variables liées, avec les complications qu'ils apportent, comme l'α-conversion ou la complexité de la notion de substitution.

On peut dans un premier temps se libérer de l'α-conversion par l'utilisation d'une formalisation alternative, où les variables sont remplacées par des "indices de de Bruijn", mais on peut aussi, comme pour le passage du λ-calcul au $\lambda\sigma$-calcul, aller plus loin et traduire les systèmes d'ordre supérieur en systèmes du premier ordre en rendant explicites les substitutions. En collaboration avec Eduardo Bonelli (Université La Plata) et Alejandro Ríos (Université de Buenos Aires), Kesner a proposé un formalisme pour spécifier des systèmes de réécriture d'ordre supérieur, appelé *Simplified Expression*

Reduction Systems à la de Bruijn ($SERS_{dB}$) [BKR00]. Ces systèmes, inspirés des *Expression Reduction Systems* [Kha90], sont plus proches des implantations des langages de programmation. Puis, dans [BKR01], les mêmes auteurs démontrent qu'un système d'ordre supérieur dans la classe $SERS_{dB}$ peut s'exprimer par un système de réécriture du premier ordre modulo une théorie équationnelle. Cette traduction est possible grâce aux substitutions explicites, et permet d'établir formellement la relation entre les systèmes de réécriture d'ordre supérieur et ceux du premier ordre. Elle permet en même temps de raisonner sur les constructeurs "de liaison" utilisés dans les langages et formalismes d'ordre supérieur.

Ces traductions ouvrent la possibilité d'exporter vers l'ordre supérieur des techniques connues dans le monde du premier ordre, dans le but soit de simplifier ou étendre des résultats connus à l'ordre supérieur, soit d'en établir de nouveaux. Parmi les questions à regarder, citons :

- la complétion dans les systèmes d'ordre supérieur, domaine dans lequel il y a déjà quelques résultats [Dug01] mais qui ne sont pas du tout fondés sur des propriétés abstraites de l'opération de substitution ;
- le *lemme des paires critiques*, donnant un critère décidable pour tester la confluence locale d'un système ;
- l'étude de la technique appelée *Dependency Pairs* [AG00] capable de démontrer la terminaison de systèmes de réécriture du premier ordre.

Réécriture d'ordre supérieur et filtrage. Les caractéristiques propres des langages ayant des opérateurs explicites comme le filtrage et la substitution sont explorées dans [CK04] à travers un formalisme appelé TPC_{ES}. L'isomorphisme de Curry-Howard est un outil central pour cette étude : aussi bien la propagation d'une substitution explicite dans un terme que le processus de filtrage de fonctions définies par cas à l'aide de motifs trouvent une interprétation très naturelle dans le processus d'élimination des coupures.

Cependant, le calcul TPC_{ES} présente un défaut majeur : il n'est pas possible d'utiliser des *symboles de fonction* ni de l'enrichir avec de nouvelles règles. Afin de pallier cette déficience, Kesner et Forest ont abouti récemment à la définition d'un cadre très général appelé ERSP (Expression Reduction Systems with Patterns) [FK03], où l'on peut représenter dans une même syntaxe des programmes fonctionnels aussi bien que des preuves par réécriture d'ordre supérieur. Ce formalisme incorpore le filtrage de manière primitive, tout en respectant les constructions de la réécriture d'ordre supérieur. Il peut être vu comme une extension des SERS (cf. plus haut) au cas de filtres complexes "à la ML".

L'avantage des ERSP par rapport à d'autres calculs réside dans le fait de

rapprocher d'une manière considérable le langage de programmation de son modèle théorique. Ainsi, dans toutes les procédures actuelles de compilation de langages fonctionnels, plusieurs mécanismes abstraits de programmation, comme par exemple la définition de fonctions par cas, sont complètement transformés et oubliés par les manipulations qui précèdent la procédure de compilation proprement dite. Ces mêmes mécanismes abstraits de programmation deviendraient tout à fait explicites dans le cadre des ERSP, où le filtrage est incorporé au niveau du formalisme calculatoire.

4.2 Isomorphismes de types

Un autre domaine dans lequel des fortes compétences en théorie de la réécriture sont nécessaires est celui des isomorphismes de types. Il s'agit de savoir décider si on peut convertir toute expression de type A en expression de type B et vice versa sans perte d'information. Si on regarde ce problème dans un langage donné, cela revient exactement à identifier les isomorphismes valables dans tous les modèles de ce langage ; par exemple, pour le λ-calcul simplement typé avec paires, l'on retrouve le problème de caractériser les isomorphismes des catégories cartesiennes fermées.

Les applications. Une procédure de décision pour ce problème a été à la base d'outils avancés de recherche de fonctions dans des bibliothèques de Caml Light et des extensions de cette procédure sont à l'étude pour les systèmes de modules d'Objective Caml. A travers la correspondance de Curry-Howard entre preuves et programmes, les isomorphismes de types deviennent des équivalences fortes de preuves qui ont été utilisées pour concevoir des systèmes de recherche de preuves dans les bibliothèques du démonstrateur Coq.

Les isomorphismes de types ont aussi des applications possibles dans la génération de code adaptatif dans des systèmes de programmation multi-langages, comme ceux proposés par exemple par le projet MockingBird de IBM Yorktown Heights. Dans cette direction, en collaboration avec François Pottier et Didier Rémy (INRIA Rocquencourt), Di Cosmo a montré par des méthodes coinductives que le sous-typage des types recursifs modulo une forme restreinte d'isomorphisme est décidable [DCPR03], ce qui pose la base théorique pour des outils de recherche dans des librairies de classes Java.

L'usage de la réécriture dans l'étude des isomorphismes de types intervient à deux niveaux. D'abord, pour prouver la complétude d'une théorie d'isomorphismes, la méthode la plus flexible et générale nécessite l'introduction d'un système de réécriture avec axiomes d'extensionnalité qui soit confluent et normalisant, sujet étudié en profondeur par Kesner et Di Cosmo

(voir par exemple [DCK94, DCK96a, DCK95]). Mais aussi, pour décider l'égalité de deux types dans la théorie, une fois que celle-ci est prouvée correcte et complète, on utilise des algorithmes faisant intervenir à nouveau la réécriture.

Le problème des sommes fortes. Balat et Di Cosmo, avec Marcelo Fiore (Université de Cambridge), ont montré dans [BDCF02] qu'en présence de sommes fortes, les isomorphismes ne sont pas finiment axiomatisables. Ce résultat a été obtenu en rapprochant cette question d'un problème de théorie des nombres énoncé par Tarski et connu sous le nom de *High school algebra problem*. La question est alors de trouver une classe de modèles plus restreinte où la décidabilité de l'isomorphisme soit assurée. Dans cette direction, Laurent montre dans [Lau03a] que si on accepte une notion de somme faible, alors la classe des isomorphismes devient finiment axiomatisable. La démonstration est sémantique, et utilise le modèle de jeux innocent de la logique linéaire polarisée.

Une autre direction consiste à se concentrer sur un système avec types récursifs, sommes et produits, mais sans la flèche, qui correspond exactement aux types de données ne pouvant pas contenir des fonctions, autrement dit, ce que l'on appelle les structures de données *passives*. Dufour et Di Cosmo ont commencé, en coopération avec Fiore (de l'université de Cambridge) une étude détaillée des isomorphismes de types dans ce cadre, et ont formulé la conjecture que la théorie des anneaux sans négatifs plus les isomorphismes définis par les types récursifs est complète pour ce système.

Isomorphismes de types et évaluation partielle. Une des conséquences de l'étude des isomorphismes en présence de sommes fortes a été la définition d'une notion de forme canonique des termes du lambda calcul simplement typé avec paires et sommes fortes, qui est obtenue à partir d'une construction catégorique sophistiquée (relations logiques de Grothendieck) d'une classe de modèles du calcul. Grâce à cette notion, il a été possible de modifier le paradigme existant de l'évaluation partielle dirigée par les types (TDPE) introduit par Danvy, en le rendant exponentiellement plus efficace en présence de types somme [BDCF04]. Ces optimisations obtenues dans le cadre d'un travail théorique sur les isomorphismes de types peuvent être réutilisées dans le domaine de l'évaluation partielle pour optimiser des programmes sans effets de bord [BD02]. Ce travail pose des questions intéressantes sur l'utilisation des opérateurs de contrôle.

Isomorphismes de types et modèles de jeux. Dans [Lau03a], Laurent a montré comment les modèles de jeux permettent une approche sémantique

de la question des isomorphismes de types. On peut ainsi caractériser les isomorphismes de types en Logique Linéaire Polarisée (pour tous les connecteurs propositionnels) et par conséquent les isomorphismes de types en logique classique pour l'appel par nom et pour l'appel par valeur, et étendre ainsi les résultats du cas intuitionniste.

Ce travail ouvre un certain nombre de questions naturelles :

– Est-il possible d'utiliser les modèles de jeux pour MLL afin de caractériser les isomorphismes de types linéaires (ce qui donnerait une preuve sémantique des résultats de Balat et Di Cosmo obtenus dans [BDC99]) ?
– Est-il possible d'étendre ce résultat à MALL((Multiplicative Additive Linear Logic) en utilisant le modèle de jeux d'Abramsky et Melliès [AM99] ?
– Est-il possible d'utiliser les résultats obtenus dans le cas polarisé à la logique linéaire non polarisée ?
– Est-il possible d'étendre ces résultats à la quantification du second ordre ?

Ce foisonnement d'activités a été couronné par un Workshop International sur les Isomorphismes de Types (WIT'2002), et un numéro spécial de la revue MSCS [DCS03].

4.3 Réécriture axiomatique.

La théorie axiomatique de la réécriture a pour enjeu de décrire les propriétés fondamentales du calcul symbolique, indépendamment de toute présentation syntaxique : causalité, concurrence, duplication, non déterminisme. La théorie est basée sur une extension simple des systèmes de transition asynchrones (ou avec indépendance) : un système axiomatique de réécriture est donné par un graphe de transition $\mathcal{G}$ muni d'une relation de permutation $f \rhd g$ entre chemins coinitiaux et cofinaux $f, g : M \to P$. Une série d'axiomes élémentaires sur $(\mathcal{G}, \rhd)$ décrit ensuite les propriétés attendues de la relation de permutation $\rhd$ dans un système de réécriture.

Les axiomes sont donnés dans [Mel02a]. Ils permettent de démontrer de manière diagrammatique plusieurs des théorèmes canoniques de la réécriture, en particulier :

1. Un théorème de standardisation [Mel02a] : on peut transformer tout chemin de réécriture en appliquant des permutations $\rhd$ en un chemin *standard* unique à permutations réversibles près.
2. Un théorème de stabilité [Mel98] qui s'énonce ainsi : si $\mathcal{V}$ désigne un ensemble de termes "résultats", typiquement les formes normales de tête dans le λ-calcul, alors il existe à partir de tout sommet M un cône $(e_i : M \longrightarrow V_i)_{i \in I}$ de chemins minimaux de M à $\mathcal{V}$: les réductions

de tête. Remarque : dans un système sans paires critiques, le cône contient une réduction de tête au plus.

3. Un théorème de correspondance pour le $\lambda\sigma$-calcul [Mel00] : toute réduction de tête du $\lambda\sigma$-calcul se projette par σ-normalisation sur une réduction de tête du λ-calcul. La normalisation des réductions *nécessaires* du $\lambda\sigma$-calcul en découle (résultat qui a été récemment étendu à tous les systèmes d'ordre supérieur par Bonelli [Bon03]).

Ces résultats approfondissent et généralisent à la plupart des systèmes de réécriture (des réseaux de Petri aux calculs d'ordre supérieurs) les résultats fondamentaux obtenus dans les années 70 et 80 pour le λ-calcul et les systèmes du premier ordre (Boudol, Huet, Lévy).

Notons au passage que les permutations $f \rhd g$ déterminent un système de réécriture 2-dimensionnel sur les chemins du graphe $\mathcal{G}$. Les propriétés de la réécriture traditionnelle, ou 1-dimensionnelle, sont donc étudiées au moyen de la 2-réécriture. De même, les axiomes imposés aux systèmes axiomatiques $(\mathcal{G}, \rhd)$ sont des "diagrammes de cube" proches d'une 3-réécriture sur des 2-termes. Il s'agit là certainement des premiers échelons d'une réécriture multidimensionnelle, qu'il reste à clarifier et à rapprocher des travaux décrits dans la section suivante.

Théorie des résidus et paires critiques. Les λ-calculs avec substitutions explicites sont pour la plupart confluents. Ils admettent cependant des paires critiques, qui posent de nombreuses difficultés techniques ou conceptuelles. Ces paires critiques sont-elles des illusions syntaxiques ? Autrement dit, est-il possible de donner une théorie des résidus dans laquelle toutes ces *obstructions* à la permutation disparaissent ? Des travaux récents de Melliès [Mel02b] suggèrent cette possibilité, en généralisant la théorie classique des résidus héritée du λ-calcul. Réaménagée, la théorie des résidus permet d'intégrer la règle d'associativité $x \otimes (y \otimes z) \xrightarrow{\alpha} (x \otimes y) \otimes z$ et son unique paire critique (notée par Huet) dont le diagramme de confluence locale est connu sous le nom de *pentagone de Mac Lane* :

$$\begin{array}{ccc} w \otimes (x \otimes (y \otimes z)) & \xrightarrow{u=w\otimes\alpha} & w \otimes ((x \otimes y) \otimes z) \\ & & \downarrow{\scriptstyle \alpha} \\ \downarrow{\scriptstyle v=\alpha} & & (w \otimes (x \otimes y)) \otimes z \\ & & \downarrow{\scriptstyle \alpha\otimes z} \\ (w \otimes x) \otimes (y \otimes z) & \xrightarrow{\alpha} & ((w \otimes x) \otimes y) \otimes z \end{array} \qquad (1)$$

Notons que ce pentagone ressemble au diagramme de permutation (ou d'homotopie) $v; u' \rhd u; v_1; v_2$ qu'on trouve dans le λ-calcul :

$$\begin{array}{ccc} \Delta(Ia) & \xrightarrow{u} & (Ia)(Ia) \\ & & \downarrow{\scriptstyle v_1} \\ \downarrow{\scriptstyle v} & & a(Ia) \\ & & \downarrow{\scriptstyle v_2} \\ \Delta a & \xrightarrow{u'} & aa \end{array} \qquad (2)$$

où $I = (\lambda x.x)$ est l'identité, et $\Delta = (\lambda x.xx)$ est le duplicateur. Pourtant, il y a une grosse différence entre (1) et (2). Dans (2), le chemin $(Ia)(Ia) \xrightarrow{v_1} a(Ia) \xrightarrow{v_2} aa$ développe l'ensemble des résidus de v après u. Dans (1) au contraire, le chemin

$$w \otimes ((x \otimes y) \otimes z) \xrightarrow{\alpha} (w \otimes (x \otimes y)) \otimes z \xrightarrow{\alpha \otimes z} ((w \otimes x) \otimes y) \otimes z \qquad (3)$$

ne développe pas un ensemble de radicaux de $w \otimes ((x \otimes y) \otimes z)$. Il faut donc revoir la théorie de Lévy, et admettre que le chemin (3) tout entier est "résidu" de l'étape v après u. On autorise donc qu'un *chemin* soit résidu d'une étape de réduction.

La théorie permet de démontrer une version forte du théorème de Church-Rosser, qui s'énonce ainsi : la catégorie $\mathcal{C}$ qui a pour objets les termes, et pour morphismes les chemins de réécriture modulo *homotopie*, i.e. modulo permutation de radicaux, a les coproduits fibrés.

Melliès montre que la théorie remaniée s'applique à la règle d'associativité comme à la présentation du monoide des tresses positives, ce qui permet de redémontrer le théorème de Garside dans l'esprit des travaux de Dehornoy [Deh04]. La théorie des résidus éclaire aussi des points laissés obscurs, et encore débattus, de la théorie de l'optimalité pour le λ-calcul, introduite par Lévy à la fin des années 1970. Finalement, ces travaux indiquent la possibilité d'une théorie des résidus dans les λ-calculs avec substitutions explicites — voir plus haut le paragraphe substitutions explicites.

Ces travaux ont reçu en 2002 le prix du meilleur article de la conférence RTA (Rewriting Techniques and Applications).

4.4 Réécriture multidimensionnelle et dihomotopie.

La réécriture multidimensionnelle développée par Burroni depuis [Bur93] offre une liaison très prometteuse entre catégories, réécriture et concurrence basée sur la structure de *n-catégorie*, extension naturelle à n dimensions des catégories (celles-ci représentant la dimension 1). Ces n-catégories ont pour générateurs les n-polygraphes, dont les entités de base sont des n-cellules.

Le recollement ou concaténation de n-cellules se fait par leurs bords de dimension $n-1$. Ces recollements qui sont d'une complexité croissante avec la dimension n, permettent la description de la réécriture sur des objets en dimension supérieure, le cas $n = 2$ correspondant à la réécriture de mots, et le cas $n = 3$ à celui de la réécriture de termes. Cela donne lieu, parallèlement, à l'étude et au développement d'une notion d'automate n-dimensionnel, où l'on retrouve, par exemple, en dimension 2 la notion classique de réseau de Petri, ou encore une extension non commutative de cette dernière.

Réécriture multidimensionnelle. Burroni dirige un groupe de travail sur des thèmes initiés dans son article [Bur93], et poursuivis dans [Bur03a], qui explore des extensions en dimensions supérieures des automates, des machines de Turing, des réseaux de Petri..., toutes ces notions étant exposées dans le cadre conceptuel unifié et géométrique des n-catégories.

En liaison avec les travaux de Goubault et Gaucher sur l'homotopie dirigée, il a donné dans [Bur03b] une version simple et naturelle des orientaux de Street susceptible de s'adapter à cette étude. Une mise au point d'algorithmes et de leur implantation pour des calculs sur ordinateur a été faite en collaboration avec Padovani.

Récemment, il a entrepris une étude de mots 2-dimensionnels, c'est-à-dire en fait d'"images", tant du point de vue des automates que de la réécriture. Pour cette étude également, l'implantation sur ordinateur est réalisée en collaboration avec Padovani.

Automates parallèles. Les recherches de Gaucher durant ces quatres dernières années ont porté sur les interactions entre l'étude des automates parallèles, d'abord avec l'algèbre homologique, puis ensuite avec la topologie algébrique tout entière.

Il y a eu deux périodes : tout d'abord une période ω-catégorique avec les publications [Gau00b] [Gau00a, Gau02a, Gau01, Gau03a], puis deux publications qui font la transition entre les deux périodes [Gau02b, GG03]. Puis la période topologique dans laquelle il se trouve maintenant [Gau03b, Gau03c, Gaua, Gauc, Gaub].

C'est la découverte au début de la première période des liens entre ses intuitions homologiques sur les automates parallèles et un modèle ω-catégorique d'automates parallèles introduits par Pratt qui lui a permis de progresser. Et c'est ensuite la découverte de la notion de flots qui lui a permis de commencer à vraiment étudier la dihomotopie, i.e. une relation d'équivalence préservant les propriétés informatiques.

L'article [Gau03a] constitue une charnière car il expose en fait les limites de

l'approche ω-catégorique des automates parallèles : elle fournit des conjectures combinatoires très dures qui sont intéressantes en elles-mêmes mais qui n'aident pas à la compréhension de la dihomotopie. La conclusion de [Gau03a] expose comment l'auteur a découvert la notion de flots, qui permet de contourner ces aspects combinatoires parasites qui empêchent d'étudier les vrais problèmes liés à la dihomotopie.

5 Thème V : Programmation

Participants : Francisco Alberti, François Armand, Vincent Balat, Sylvain Baro, Anne-Gwenn Bosser, Emmanuel Chailloux, Guy Cousineau, Roberto Di Cosmo, Yves Legrandgérard, Li Zheng, Jean-Vincent Loddo, Pascal Manoury, François Maurel, Micaela Mayero, Alexandre Miquel, Raphaël Montelatici, Jean-Marie Rifflet, Paul Rozière, Jérôme Vouillon.

Le thème *programmation* regroupe un large spectre d'activités liées à la mise en œuvre d'outils logiciels. La part essentielle de cette activité est un apport au développement des langages de programmation, par proposition :

- de nouveaux langages, de nouveaux compilateurs,
- d'extensions de langages existants ou d'interfaçage entre différents langages,
- d'outils de certification de programmes.

Mais les compétences ou l'expertise des membres du laboratoire impliqués dans les activités du thème programmation les amènent également à développer ou mettre à disposition de la communauté des outils et systèmes originaux ou innovants plus généraux, citons : outils de synchronisation de fichiers, protocole IPv6, système de fenêtres XFree86, etc.

Concrètement, l'activité du thème Programmation exploite en premier le paradigme pleinement fonctionnel à types paramétrés du langage Objective Caml[2] (OCaml) aussi bien comme vecteur de recherche pour la partie conception, implantation et extension de langages que comme objet d'étude (typage, preuves de programmes). Il est utilisé comme langage d'implantation dans la majorité des réalisations. Cependant, l'activité du thème Programmation ne connaît pas d'exclusive et concerne aussi :

- l'implantation de machines virtuelles de Java (JVM) ou de .NET,
- le lien avec les arbres XML pour les systèmes de types,
- des réalisations dans d'autres langages comme Python, ou l'étude de la concurrence dans les systèmes d'exploitation (Unix et Chorus).

[2] `http://caml.inria.fr/`

Nous décrirons l'aspect développement du thème selon trois principaux axes : langages (section 5.1), assistants de preuves (section 5.2), et programmation concurrente, répartie et parallèle (section 5.3). Nous terminerons en soulignant un second aspect de l'activité développée par le thème Programmation qui est son souci de *diffusion des connaissances ou des outils logiciels* "hors du monde purement académique" (section 5.4).

5.1 Langages

Nous décrivons d'abord nos contributions en conception et implantation de langages, puis celles de développement de bibliothèques et d'extensions, et enfin concernant l'analyse statique des programmes.

XDuce. Vouillon et Di Cosmo ont lancé le projet Ocamlduce, en collaboration avec Michel Mauny (INRIA Rocquencourt). Ce projet vise à rapprocher XDuce[3], un langage de manipulation de données XML, d'OCaml. L'objectif est de profiter depuis XDuce des bibliothèques existantes écrites en OCaml et des performances du compilateur de ce langage. Ce projet a amené Vouillon à étudier des extensions du système de types de XDuce avec des enregistrements, des fonctions de première classe et du polymorphisme. Des questions de différents ordres se posent. L'algorithme de sous-typage doit rester rapide en pratique, bien que sa complexité théorique soit exponentielle. Vouillon a implanté des heuristiques qui permettent d'éviter cette explosion la plupart du temps, et qui semblent très bien fonctionner en pratique. Une autre difficulté d'ordre plus théorique est la spécification des règles de sous-typage (voir thème III).

Isomorphismes de types. L'étude des isomorphismes de types, tel que décrit au thème IV, fournit la base d'applications de recherche en librairies fonctionnelles ou objet. En collaboration avec Boris Yakobowski (stagiaire ENS Lyon), Di Cosmo étudie l'extension de ces techniques aux langages de modules, tel le système implanté dans Ocaml. Un premier prototype est en phase de test. En collaboration avec Didier Rémy et François Pottier (INRIA), Di Cosmo étudie aussi la possibilité de réaliser un système de recherche de classes Java à grande échelle.

Activités autour de micro-noyaux. Rifflet a participé au portage de la machine virtuelle Java (JVM) pour le système Chorus, dont le multithreading temps réel facilite l'implantation de la partie concurrente de la JVM.

[3] `http://xduce.sourceforge.net/`

En collaboration avec François Armand (chef de projet dans la start-up Jaluna créée par les anciens de Chorus, et recruté comme PAST à Paris 7), il entreprend par ailleurs un travail de synthèse autour des machines virtuelles dans le domaine des systèmes d'exploitation (user mode Linux, xen, adeos, Jaluna2, etc.).

Compilation vers `.NET`. La nouvelle plate-forme `.NET` de la société Microsoft apporte un haut niveau d'interopérabilité entre composants développés dans différents langages, mais autorise aussi grâce aux initiatives "open source" (`sscli`, `mono`) d'être multi-plateformes. La compilation d'OCaml vers cette plate-forme a débuté à l'été 2002. Le travail consiste tout d'abord à développer le prototype d'un compilateur du langage OCaml vers le bytecode MSIL utilisé dans l'environnement Microsoft .NET. Il s'agit ensuite d'étudier l'interopérabilité entre un langage fonctionnel tel que OCaml et d'autres langages portés vers cette machine, en termes de faisabilité, de performance et d'ergonomie. Un premier prototype non optimisé, appelé `ocamil`[4][CMP04], a été réalisé par Montelatici et a permis d'initier une collaboration avec la société *Lexifi*[5] qui a pris la forme d'un stage co-encadré durant l'été 2003.

Toplevel embarqué. Clément Capel (stagiaire de maîtrise Paris 6) et Chailloux ont étudié les modifications à apporter au compilateur OCaml pour embarquer le toplevel OCaml sous forme d'une bibliothèque (DLL) dans une application C. L'intérêt est alors de pouvoir compiler et exécuter des programmes OCaml à la volée, provenant d'une autre source, ou engendrés par l'applicaton elle-même. Une première application a été d'adapter un *plugin* pour le navigateur Netscape pouvant exécuter du code OCaml dans sa fenêtre. Ce travail se poursuit avec l'utilisation d'OCaml comme langage de script embarqué dans VBA-Excel[CCE04].

Programmation et interfaces graphiques. Cousineau a prolongé les travaux qu'il avait menés avec Chailloux sur la bibliothèque graphique MLgraph en développant des applications permettant de réaliser des pavages. Une de ces applications est décrite dans [Cou02]. Un travail a récemment été entrepris avec Di Cosmo dans le cadre du projet CaraML (voir plus loin) pour en donner une version parallèle. Une autre application, en cours d'implantation avec Vincent Ripoll (étudiant de DESS), est un éditeur graphique de motifs de pavages hyperboliques. Cette activité orientée vers les pavages, a par ailleurs conduit Cousineau vers des questions de nature plus théorique

[4] `http://www.pps.jussieu.fr/~montela/ocamil`
[5] `http://www.lexifi.com`

comme la caractérisation de motifs associés à certains groupes de symétrie [Cou00].

Par ailleurs, au départ dans le cadre d'un cours de DESS, Cousineau a investi dans la connaissance du système MacOSX [Cou99]. Une application OCaml pour MacOSX incluant la bibliothèque graphique est en cours de développement en collaboration avec Chailloux.

Chroboczek est co-développeur du système de fenêtres XFree86, un système orienté objet à grande échelle. Il s'occupe en ce moment de la transition du sous-système gérant les polices vers des formats plus modernes. La documentation de sa contribution est accessible en ligne[6].

Avant de rejoindre le laboratoire, Miquel a créé le moteur d'affichage MLdvi pour les fichiers DVI produits par le logiciel TEX ; ce travail a ensuite été étendu par le groupe CRISTAL de l'Inria à un logiciel de présentation plus complexe, ActiveDVI. Di Cosmo a coopéré très activement à son développement pendant son détachement, et continue de le faire.

Extensions objet. Parce que le polymorphisme introduit par l'extension objet du langage OCaml est un polymorphisme de rangées, de nombreux programmeurs objet sont décontenancés par la relation de sous-typage induite, et de plus certains modèles de conception (Design Patterns) objet ne peuvent pas être implantés. Pour pallier cette limitation, une extension `cocaml`[7] a été réalisée [Cha02a]. Elle autorise le "downcast" d'un objet entre classes en relation d'héritage et de sous-typage. Ce travail a été étendu pour un modèle de persistance sûr [Cha02b].

Extensions avec des templates. Une extension du langage Objective Caml permettant de générer du code à la compilation à partir d'informations de types a été proposée par Maurel. Un prototype est accessible[8] et une soumission est en cours [Mau03b]. Cette extension permet par exemple de générer des itérateurs arbitraires à partir de définitions de types.

Interopérabilité. Dans le but de construire des applications multi-langage, Grégoire Henry (stagiaire de maîtrise Paris 6) et Chailloux ont défini un IDL (langage de définition d'interface) simple pour la description des classes Java. La communication d'OCaml vers Java est directe. La communication de Java vers OCaml passe par un mécanisme de rappel. L'outil de génération de code, appelé O'Jacare [CH04], apporte l'automatisation des codes d'encapsulation. Une première application a été d'implanter la visionneuse MLdvi de Miquel

[6] `http://www.xfree86.org/current/fonts.html`

[7] `http://www.pps.jussieu.fr/~emmanuel/Public/Dev/coca-ml/`

[8] `http://www.pps.jussieu.fr/~maurel/programmation`

sur le patron de conception Modèle-Vue-Contrôleur où la partie algorithmique du Modèle est en OCaml et l'interaction avec l'utilisateur en Java. Ce travail se poursuivra en l'adaptant à la communication entre `ocamil` et C# qui possèdent une bibliothèque d'exécution commune (en .NET).

Analyse statique. Des optimisations pratiques, comme l'"inlining" ou l'évaluation stricte, peuvent être justifiées lorsqu'on découvre comment une valeur est "utilisée" dans un contexte donné. Cette idée semble maintenant relativement acceptée. Dans sa thèse, Alberti présente un cadre théorique général d'analyse statique pour l'inférence de propriétés d''usage' ou, *propriétés structurelles* (en référence à la logique linéaire) des programmes fonctionnels. Il formule un système de typage à la Church pour un langage intermédiaire dans le style de PCF, mais comportant des annotations structurelles. Le problème de l'analyse statique consiste alors à trouver une traduction du langage source vers ce langage intermédiaire. On montre que l'on peut voir toutes les traductions possibles comme les solutions d'un ensemble d'inéquations appropriées, dont la plus petite solution correspond à la traduction la plus précise ou optimale. Comme le prouve le prototype récemment implanté, l'inférence des propriétés structurelles pour un langage réel est relativement simple et efficace.

Alberti étend ses analyses au sous-typage et au polymorphisme d'annotations. Ce dernier, qui est un mécanisme d'abstraction sur des annotations, est une extension clé dans la pratique, car il permet à l'analyse de garder son expressivité en présence de modules compilés séparement.

Alberti prouve plusieurs propriétés standards pour l'ensemble des systèmes de typage, ainsi que leur correction sémantique par rapport à la sémantique opérationnelle du langage source.

5.2 Assistants de preuves

L'étude menée tourne autour de deux axes :

- la conception et la présentation d'un système logique intuitif qui ne sacrifie rien à la rigueur formelle, gage de sûreté ;
- la définition d'une architecture logicielle du *moteur de preuve* qui permette le dialogue avec une interface graphique contemporaine.

Le premier élément concret de cette étude sur les systèmes de preuve est son architecture logicielle. Il a mené à la réalisation du système décrit ci-dessous. Mais on y trouve également des résultats reposant sur l'utilisation d'autres systèmes existants.

Assistant de preuves pour ML. Baro a développé un assistant de preuve autonome, appelé `PAF !`, ayant pour but la vérification de programmes fonctionnels en ML. Son manuscrit de thèse [Bar03] décrit de façon exhaustive tous les aspects de ce programme développé en OCaml :

- le formalisme logique sous jacent ;
- la spécification du système de preuve et de son interface graphique ;
- l'implantation fondée sur une architecture originale mettant à profit aussi bien la programmation objet que fonctionnelle.

Dans [BM03a], Baro et Manoury présentent et donnent la preuve de correction du noyau du système formel dédié à la preuve de terminaison des programmes.

Interface pour les assistants de preuves. Legrandgérard a finalisé la spécification du protocole de communication PEP (Proof Engine Protocol) et en a donné deux implantations (l'une en Python et l'autre en C++). PEP est un protocole binaire au-dessus de TCP permettant l'échange de données entre un moteur de preuves et une interface d'édition interactive de théories et preuves formelles qu'il a commencé à spécifier en juin 2000. Une implantation opérationnelle de cette interface, écrite en Python/TK, a permis son interconnexion avec le moteur de preuves développé par Baro et une démo de l'assistant de preuves ainsi obtenu a été présentée à *TYPES 2003*.

Assistant de preuves en mathématiques. Rozière collabore plus particulièrement avec Christophe Raffalli (Université de Savoie), auteur du système d'aide à la preuve PhoX[9]. Il a réorganisé et complété certaines des librairies de PhoX. Il a fourni quelques exemples ; en particulier une formalisation d'une preuve de consistance relative de la théorie des ensembles dans la théorie des ensembles sans extensionnalité (cf. thème III). Il participe au développement du mode emacs pour PhoX que l'on trouve avec la distribution de ProofGeneral[10]. Il participe également à la rédaction de la documentation du logiciel [RR02] et à celle d'un polycopié d'introduction [DRR03]. Enfin, dans le cadre de l'utilisation des assistants de preuve en mathématiques, Rozière travaille sur la *formalisation de la géométrie* en PhoX. Le but est d'ajouter suffisamment d'automatismes pour que des preuves de géométrie élémentaire s'écrivent de façon naturelle avec un assistant de preuves.

[9] `http://www.lama.univ-savoie.fr/~RAFFALLI/phox.html`
[10] `http://www.proofgeneral.org`

Preuves en calcul formel. Mayero a poursuivi un travail initié lors de son post-doc dans l'équipe Programming Logic à l'Université de Chalmers à Göteborg (Suède) sur l'interaction entre le calcul formel et les preuves formelles. Un *mode Maple* pour Coq[11] permet d'utiliser la puissance de MAPLE pour calculer ou faire des simplifications, et montrer, grâce à Coq, que le résultat rendu est correct [DM02].

5.3 Programmation parallèle et répartie

Chailloux et Di Cosmo s'intéressent au dévéloppement d'applications parallèles massives coordonnées par des langages fonctionnels, et ils participent activement au projet ACI Grid *CaraML*[12], qui regroupe les chercheurs français intéressés à ces thématiques.

Caml-Flight. Chailloux et Christian Foisy (Digital Fountain, Fremont, Californie) ont poursuivi leur travail d'extension data-parallèle de Caml-light débuté en 1995. Le prototype de l'époque, Caml-Flight, conservait la propriété de déterminisme pour les programmes fonctionnels parallèles. Caml a évolué depuis en offrant un mécanisme de processus légers, un modèle de persistence ainsi que l'outil `camlp4`[13] pour les extensions de syntaxe du langage. Ces outils ont facilité la construction d'une nouvelle implantation très concise et portable appelée `ocf`[14] [CF03a, CF03b] qui étend la conservation du déterminisme pour les programmes parallèles fonctionnels et impératifs.

OCamlp3l. Le projet `OcamlP3l`[15], né d'une collaboration antérieure avec l'INRIA et l'Université de Pise, vise la réalisation d'une librairie de squelettes de parallélisation pour l'exécution de programmes sur une machine multiprocesseurs ou sur un ensemble de machines à travers un réseau. Les squelettes de parallélisation peuvent être vus à la fois comme des combinateurs classiques tels que `map` ou `reduce` et comme des constructions typiques de parallélisme dont la compilation sur une structure de machine ou de réseau peut être réalisée de façon systématique. A travers ces "combinateurs de parallélisme", le programmeur peut exprimer le parallélisme exploitable dans ses programmes sans avoir à se soucier des aspects de synchronisation qu'implique la programmation du parallélisme. Ces aspects sont pris en charge par le compilateur.

[11] `http://coq.inria.fr`
[12] `http://www.caraml.org`
[13] `http://caml.inria.fr/camlp4`
[14] `http://www.pps.jussieu.fr/~emmanuel/Public/Dev/`
[15] `http://www.ocamlp3l.org`

Les applications sont nombreuses. Le projet ESTIME de l'INRIA Rocquencourt utilise `OcamlP3l` pour la coordination de solveurs numériques préexistants [CVDCW03], Cousineau propose de s'en servir pour paralléliser la génération de pavages hyperboliques, et Di Cosmo étudie la compilation efficace d'opérations data-parallèles sophistiquées [DCP03] en coopération avec Susanna Pelagatti (Univ. de Pise) et Li.

Le reste de cette section est consacré à la *répartition*, abordée tant sous son aspect fondamental (protocoles, systèmes) que sous son aspect applicatif.

Protocole IPv6. Legrandgérard est impliqué dans le développement et la mise en œuvre du nouveau protocole internet IPv6 en France depuis 1996, date à laquelle il a rejoint le G6 dont il héberge le site WEB au laboratoire PPS[16]. Il participe également fortement au projet pilote IPv6 Renater 2, initié au début des années 2000, qui a donné naissance à un réseau IPv6 natif dans le cadre de Renater 3 en octobre 2002. Il a, en collaboration avec B. Tuy (en charge du projet IPv6 à Renater) procédé à la migration du PIR Île de France vers ce nouveau réseau IPv6 natif (IPv6/ATM) ainsi qu'à l'établissement d'un nouveau plan d'adressage IPv6, rendu nécessaire par l'allocation par RIPE des premiers préfixes de production IPv6, qui a été repris presque à l'identique dans le cadre de Renater 3.

Unison : synchronisation de fichiers `Unison`[17] est un programme de synchronisation de fichiers entre deux ordinateurs. Vouillon a modifié ce programme afin qu'il soit capable de transférer simultanément plusieurs fichiers. Les performances en ont été ainsi grandement améliorées. Par ailleurs, il a défini avec Benjamin Pierce (University of Pennsylvania) une spécification formelle d'un programme de synchronisation de fichiers, précisant les modifications qu'un tel programme était autorisé à faire. À son arrivée à PPS, il a poursuivi ce travail et en a validé la spécification en montrant la correction d'une version simplifiée de `Unison` à l'aide de l'outil d'aide à la preuve `Coq`[18] [PV03].

Lors du développement d'`Unison`, Jérôme Vouillon a été amené à écrire une bibliothèque d'expressions rationnelles basée sur des automates déterministes. En effet, les bibliothèques existantes s'avèrent beaucoup trop lentes sur de grosses expressions rationnelles. Il a récrit et grandement amélioré cette bibliothèque, nommée RE[19]. Alors que la version initiale utilisée par `Unison` était très limitée (elle permettait juste de savoir si une chaîne de

[16] `http://www.ipv6.pps.jussieu.fr`
[17] `http://www.cis.upenn.edu/~bcpierce/unison`
[18] `http://coq.inria.fr/`
[19] `http://www.sf.net/projects/libre/`

caractères est reconnue par un motif), cette nouvelle version permet de rechercher un motif dans une chaîne de caractères, de savoir à quel endroit se trouve ce motif, ainsi que des sous-motifs le composant, dans la chaîne. C'est l'une des seules bibliothèques à base d'automates déterministes qui permette cela.

Applications de type Monde Virtuel réparti. Il s'agit ici de recherches menées dans le cadre du projet RNTL *EDICA*[20], qui réunissait initialement deux laboratoires publics (PPS avec Chailloux et le LIP6 avec Philippe Codognet) et la société *Cryo-Networks*, qui développait un langage propriétaire (SCOL) destiné à la réalisation d'applications de type Monde Virtuel réparti. Suite au désistement du partenaire industriel (cessation de ses activités `R&D`), le projet a repris avec la société *Virtools*. Désormais, le projet est surtout axé sur les aspects architectures et modularité du serveur d'une application de type monde virtuel, et donnera lieu à la réalisation d'un prototype d'architecture serveur. Dans la première phase du projet, un prototype du langage SCOL, `scocaml` [EDI02], a été construit pour pouvoir tester les futures extensions. Il utilise l'outil `camlp4` (cf. plus haut) en compilant un programme SCOL vers un programme OCaml. Cette traduction montre les différences entre les deux langages, en particulier dans leur système de types. Le langage a été enrichi d'un mécanisme d'exceptions et d'un système de modules [EDI03].

Les concepteurs d'applications massivement multi-joueurs n'ont pas les outils pour une bonne analyse des ressources nécessaires tant du point de vue architecture répartie que de la persistence et de la sécurité. Ici, l'expérience d'Anne-Gwenn Bosser de par ses deux années passées à la `R&D` de Cryo-Networks complète les compétences de PPS sur la répartition et les langages de communications [BA03].

5.4 Diffusion

Cette activité est importante et multiforme :

- contributions actives à la création et à la diffusion du logiciel (cf. plus haut la participation de Chroboczeck au projet Xfree86 et de Legrandgérard à la normalisation IPv6, voir aussi l'activité DemoLinux ci-dessous), ainsi qu'aux réflexions sur l'impact du logiciel libre dans notre société [DC03] ;
- contacts industriels (Cryo-networks, Virtools, Jaluna, Lexifi, cf. plus haut) et formation permanente (voir ci-dessous) ;
- rédaction d'ouvrages d'audience générale sur OCaml, Unix, Chorus et Ipv6 (voir plus bas).

[20] `http://www.telecom.gouv.fr/rntl/AAP2001/Fiches_Resume/EDICA.htm`

DemoLinux. Ce projet, lancé par Di Cosmo avec la collaboration de Balat et Loddo, vise à promouvoir le système d'exploitation GNU Linux et le logiciel libre par le biais de disques CD ROM auto-configurables et fonctionnant sur les principales architectures PC IBM ou compatibles. Cette initiative a donné lieu à de nombreuses collaborations avec le Ministère de l'Education Nationale (CNDP), le Département Général pour l'Armement (DGA), Sun Microsystems France, Red Hat France, Mandrake Soft, Linbox. Le CD ROM est diffusé conjointement avec l'INRIA, ainsi qu'à plusieurs reprises depuis 1999 par des magazines de la presse informatique spécialisée en France et à l'étranger (entre autres : Italie, Thaïlande et Japon). Il a été présenté lors de la 1ère Journée Denis Diderot du Libre qui s'est tenue à Paris 7 le 28 Novembre 2001.

Formation permanente. Chailloux, Manoury et Cousineau ont donné plusieurs cycles de formation permanente[21] sur OCaml à des industriels provenant principalement du CEA et de France-télécom `R&D`. Depuis 1999, six sessions d'une semaine ont été prodiguées dont deux sur le site de Lannion de France-Télécom. Ces interventions permettent de créer et de maintenir les contacts avec ces entreprises [BC00].

Livres. Sur la période de référence, trois livres reflètent la volonté de diffusion pour un public plus large des problématiques de ce thème.

- Rifflet, en collaboration avec Jean Baptiste Yunes (LIAFA), a intégré en un seul volume de 796 pages, en les faisant fortement évoluer, ses deux anciens livres de référence sur Unix [RY03].
- Legrandgérard participe au collectif "Gisèle Cizault" qui produit l'ouvrage de référence sur Ipv6 [Ciz03].
- Chailloux, Manoury et Bruno Pagano (Esterel Technologies) ont écrit 'Développement d'Applications avec Objective Caml' [CMP00]. La version complète est en ligne[22]. Un groupe de volontaires en a effectué une traduction anglaise, qui est également en ligne[23]. Ces versions ont été mises dans le CD-ROM de l'Unesco 'Construire le Cyberspace'. Toutes ces diffusions sont effectuées avec l'accord de l'éditeur.

Un autre livre aurait dû voir le jour en version papier : 'Programming under ChorusOS' [Rif02] chez SUN Press, mais suite au désengagement de la société SUN dans Chorus, la publication a été arrêtée. Néanmoins, la version intégrale de 596 pages est accessible[24]. Enfin, trois livres sont en préparation :

[21] `http://wwwadm.admp6.jussieu.fr/fp/INFO2003/Stages`
[22] `http://www.oreilly.fr/catalogue/ocaml.html`
[23] `http://caml.inria.fr/oreilly-book/`
[24] `http://www.pps.jussieu.fr/~rifflet/book4.html`

– Un livre introductif au langage Scheme comme premier langage de programmation est en cours de finition. Il est écrit par un collectif constitué d'Anne Brygoo, Titou Durand, Pascal Manoury, Christian Queinnec et Michèle Soria. Cet ouvrage a l'originalité de reposer sur l'expérience d'innovations pédagogiques d'auto-entrainement et de corrections automatiques présentées dans [BDM+02b, BDM+02a, QC02].
– Rifflet attaque avec François Armand l'écriture d'un livre sur la programmation temps-réel avec en particulier tout ce qui concerne le système Jaluna.
– Le polycopié 'Interfaces Visuelles' [Cou99] de Guy Cousineau fournira la base d'un livre prévu pour fin 2004.

6 Thème VI : Logique, Concurrence et Modélisation

Participants : Marc Chiaverini, Vincent Danos, Deng Yuxin, Fabien Tarissan.

Les systèmes d'information d'aujourd'hui se distinguent par leur dense réseau d'interconnexions et leur exposition à un monde ouvert peuplé de processus aux comportements mal définis. On est loin du calcul distribué des années 70 où il ne s'agissait que de répartir une unique tâche de calcul sur un réseau de machines à la topologie bien spécifiée.

On parle de calcul *ubiquitaire* pour suggérer la variété des supports de calculs, *mobile* pour évoquer les modifications dynamiques des structures de communications et *ouvert* pour souligner la nature changeante de l'environnement de calcul, et le besoin subséquent d'identifier les partenaires de communication et de négocier dynamiquement leur mode d'interaction.

Comme c'est l'habitude en informatique théorique, cette poussée de la pratique du calcul pose de sérieux défis à la théorie : comment parler de spécification, de sécurité, de garantie sur l'utilisation des ressources, de négociations de droits de migration, voire d'adjudication de débits sur des flots de données ?

L'avènement de ces nouveaux systèmes de calcul répartis ou décentralisés ne fait que rendre plus pressant le besoin de fondements mathématiques, car seuls ces derniers sont justiciables d'études sémantiques et peuvent à terme infléchir la conception des langages et des architectures de la nouvelle ère du "global computing" dans la direction d'une meilleure cohérence[25] et donc

[25] Pour ne citer qu'un exemple, PHP, un des langages couramment utilisés dans la programmation dynamique de pages web, est fondé sur une représentation de l'interaction client/serveur qui est incohérente, mais surtout qui n'est même pas substanciée par un formalisme. La modélisation de la programmation web interactive est aujourd'hui un sujet de recherche en plein essor (cf. les travaux de Felleisen, Thiemann).

d'une meilleure maitrise des coûts liés à la sécurité et au developpement en général.

L'objet de la théorie de la concurrence[26] est précisément de proposer des systèmes formels qui permettent d'idéaliser des situations de communications et de fournir les moyens de les analyser dans le but d'en prouver le bon fonctionnement. Au-delà des formalismes de base proposés par Milner, CCS et le π-calcul, et de la gamme de leurs différentes extensions, on trouve le calcul des Ambiants Mobiles dû à Cardelli et Gordon (qui a déjà lui aussi de nombreuses variantes), qui cherche à capturer formellement la notion de domaine dynamique de calcul.

Dans le laboratoire, les recherches dans ce domaine ont débuté récemment, et ont emprunté quatre directions principales : l'étude des problèmes de terminaison dans les calculs concurrents (section 6.1), l'étude des fondements mathématiques des systèmes concurrents probabilistes (section 6.2), la mise au jour de structures logiques sous-jacentes à la programmation concurrente (section 6.3), et les applications potentielles de la théorie de la concurrence à la représentation et l'analyse des systèmes biologiques (section 6.4). Ce dernier axe en émergence propose à la théorie de la concurrence un champ d'application différent du champ traditionnel (évoqué ci-dessus) et renouvelle vigoureusement l'étude de ses formalismes, comme nous le verrons.

6.1 Systèmes concurrents et Terminaison

Les questions de terminaison ont surtout été étudiées dans le cadre de la réécriture et du λ-calcul, et ont jusqu'alors été négligées dans les calculs de processus, qui modélisent en effet principalement des processus sans fin comme le web ou les systèmes d'exploitation. Mais précisément, dans ces applications, on souhaite que certains sous-systèmes terminent. On veut par exemple être assuré qu'une interrogation sur un moteur de recherche aboutisse. Davide Sangiorgi (Université de Bologne) [San01] a montré la terminaison d'un premier sous-ensemble du π-calcul par une méthode de réalisabilité, et Deng travaille actuellement sur d'autres fragments qui se prêtent à des méthodes de preuves de terminaison plus combinatoires, et qui couvrent des exemples suffisamment expressifs (tables de symboles, codages du λ-calcul).

[26] On utilise ici, comme c'es devenu l'habitude, le mot "concurrence" au sens anglais du terme, qu'il faudrait plutôt traduire par "parallélisme".

6.2 Fondements mathématiques des systèmes concurrents probabilistes

Ces dernières années ont vu les probabilités entrer en force dans le domaine de la concurrence. Il est important d'identifier les structures mathématiquement idoines pour décrire les calculs probabilistes, d'autant que dans ce cas précis ces structures existent et ont été obtenues par un long travail mathématique qu'on n'a pas envie de refaire.

Un des modèles de base sur lequel la recherche moderne se penche depuis quelques années est celui de Processus de Markov étiqueté (noté ci-dessous LMP, pour "Labelled Markov Processes"). Les LMP sont des systèmes de transition probabilistes où l'espace d'états est un espace mesurable abstrait, incluant en particulier le cas d'ensembles continus d'états ; ils forment un modèle de calcul simplifié basé sur une notion d'interaction par synchronisation sur des étiquettes, dont l'intérêt est qu'il permet de décrire des situations probabilistes interactives tout en restant suffisamment simple pour qu'il soit possible d'en mener une étude mathématique et de déployer des techniques automatiques de vérification (model-checking) dans les bons cas. Un des résultats remarquables de cette montée en généralité (si on compare aux traditionnels automates probabilistes) est la découverte par Josée Desharnais (Université Laval, Québec) et Prakhash Panangaden (Université McGill, Montréal) qu'une logique temporelle purement positive caractérise la bisimulation forte entre de tels processus.

Danos et Desharnais ont donné une extension de cette logique de bisimulation à l'aide de points fixes qui se marie très bien avec la théorie de la mesure et permet de formuler des propriétés cycliques, stables à travers le temps [DD03a]. En collaboration avec Panangaden, ils ont montré comment on peut dériver de cette approche logique une notion flexible d'approximation finie de LMP arbitraire [DD02, DD03b, DDP03] basée entre autres choses sur l'utilisation d'espérances conditionnelles, un outil central de la théorie mathématique des probabilités. Ces techniques sont d'un intérêt qui excède le simple modèle des LMP. Elles proposent une notion d'approximation qui pour la première fois en informatique théorique ne tient pas dans le paradigme de Scott, en ce que la notion sous-jacente de convergence en information n'est pas directement fondée sur une notion d'ordre, mais indirectement sur la notion de filtration (d'usage courant en probabilités) et la notion cousine de martingale. On peut escompter des répercussions théoriques intéressantes sur les fondements de la sémantique dénotationnelle (la théorie des domaines) ainsi que sur le concept de base d'équivalence de processus, à savoir la bisimulation.

Par ailleurs, le fait de travailler avec des espérances conditionnelles (essentiellement des moyennes sur des classes d'équivalence de propriétés temporelles)

devrait permettre de faire des progrès sur la robustesse des approximants. On en attend aussi des applications en apprentissage, spécifiquement en algorithmique des processus de décision Markoviens. Ceux-ci sont des LMP finis mais a priori très gros, munis d'une notion de récompense que l'apprentissage a pour but de maximiser en moyenne ; l'espoir est que les approximations basées sur l'espérance conditionnelle fournissent des réductions du nombre d'états rendant possible le calcul approché de stratégie optimale.

6.3 Structures logiques de la programmation concurrente

Depuis quelques années, les études des calculs de communication tendent à abandonner les méthodes directement syntaxiques basées sur les systèmes de transition étiquetés qui ont fait les beaux jours de l'étude de CCS, pour se pencher de plus en plus vers une analyse compositionnelle donnée par des systèmes de typage. La raison en est qu'un typage est au fond une représentation explicite, plus ou moins détaillée, des propriétés de comportement des processus. S'assurer de propriétés en les attachant explicitement aux constructions de base de la syntaxe est l'approche la plus naturelle lorsque les langages de modélisation deviennent complexes. Et c'est cette approche naturelle qui semble emporter la faveur de la communauté en ce moment. Cependant ! La très grande majorité de ces systèmes, pour des raisons historiques, sont basés sur des syntaxes qui ne représentent pas les calculs d'ordre supérieur (c'est-à-dire où les processus peuvent recevoir et envoyer des processus et non pas simplement des références à des processus).

Prenant le contre-pied de cette tendance, Danos définit un modèle de communication [DG03], qui s'apparente au π-calcul d'ordre supérieur étudié par Sangiorgi et Thomsen, mais qui est surtout la généralisation naturelle du λ-calcul avec appel-par-nom à la communication. Ce langage est muni d'un système de typage dont les types dépendent de noms de canaux qui eux-mêmes sont valués dans une algèbre axiomatique de privilèges. Les axiomes sont choisis de sorte que tout modèle de typage ait la propriété que le type d'un processus ne peut que décroître durant son exécution. Du point de vue applicatif, il faut comprendre le type d'information que ce théorème de préservation du typage permet d'obtenir, et le sens qu'on peut lui donner en sécurité. Il serait intéressant de comparer ce système avec les "types comportementaux" définis par Yoshida. L'approche sémantique, avec des types paramétrés dans un modèle de privilèges, semble riche de possibilités.

Du point de vue théorique et dans la mesure où le système est une extension naturelle du λ-calcul, sur lequel on sait aujourd'hui beaucoup de choses, il faudrait comprendre comment adapter l'outil de réalisabilité classique développé par Krivine (voir thème 4). Cet outil s'est révélé très efficace dans le cas séquentiel et notamment pour étudier de manière souple des ex-

tensions syntaxiques inhabituelles du contrôle [BD03]. Dans la mesure où l'on peut concevoir la communication comme une extension du contrôle par exceptions [DK00] dans un environnement de calculs concurrents, il n'est pas fou d'imaginer une telle exportation des techniques Kriviniennes.

6.4 Modélisation des systèmes de la biologie moléculaire

Commençons ici par donner quelques éléments de contexte.

La révolution post-génomique Il faut rappeler tout d'abord que dans une large mesure l'histoire de la biologie moderne est l'histoire de l'évolution des techniques d'observation du vivant. D'aucuns considèrent que nous vivons pas moins qu'une révolution technologique dans ce domaine. De nouveaux procédés expérimentaux, tels que les puces à ADN, le double hybride et la spectrographie de masse, permettent l'observation des systèmes biologiques jusqu'au niveau des macromolécules biologiques (protéines, ARN, ADN) et de leurs interactions avec une largeur d'observation qui peut recouvrir un organisme tout entier. Pour la première fois en biologie moléculaire, il semble qu'une approche "top-down" qui seconde l'approche usuelle soit possible [ITR+01]. Les instantanés de l'état moléculaire de la cellule qui sont maintenant disponibles pourraient se révéler précieux pour la reconstruction et la validation des mécanismes moléculaires sous-jacents aux activités de la cellule. Dès lors, la question fondamentale du langage dans lequel représenter ce qu'on essaye de reconstruire se pose. Quel formalisme doit-on utiliser pour décrire, analyser et simuler ces réseaux denses d'interactions qui composent la cellule ?

La majeure partie des travaux présents de reconstruction se préoccupe de réseaux de régulation transcriptionnelle. Ceux-ci sont des graphes qui représentent les influences entre les gènes et il n'est en effet pas nécessaire ici de s'interroger sur le formalisme car la situation est simple. Mais tôt ou tard, de nouveaux niveaux de détails entreront en ligne de compte, tels que les interactions entre protéines ou les interactions protéine-ADN. Des notations graphiques variées existent déjà en biologie moléculaire comme la base de données de voies de signalisation KEGG [KEG], mais le travail de reconstruction réclame un accès à la dynamique qui sous-tend ces réseaux. Conséquemment, les langages destinés à représenter le détail moléculaire des interactions intra-cellulaires devront pouvoir décrire cette dynamique.

Comme on s'en doute bien, cette question n'est pas passée inaperçue et différents langages ont été proposés : les systèmes différentiels ordinaires [SEJGM02], des réseaux de Petri de différentes sortes [HT98, MDNM00], les automates hybrides [GT01] et les "State Charts" [KCH01] pour en citer quelques-uns. Tous ces langages donnent en effet les moyens de contruire des

modèles, mais on peut remarquer qu'ils ne portent pas en eux une description des processus élémentaires et des objets de la biologie moléculaire.

Modélisation en π-calcul. Récemment, Regev, Shapiro [RSS01, PRSS01, RS02] et un petit nombre d'autres auteurs [BDP02, DL03c] ont mis en avant l'idée que les algèbres de processus pourraient être de bons langages de formalisation de la biologie moléculaire. Cette proposition se distingue nettement des précédentes dans la mesure où elle tente d'expliquer à un certain niveau d'idéalisation ce qu'est la biologie moléculaire. Par exemple, en π-calcul, une molécule est un agent, la liaison est une communication, et le contact physique continu est représenté par le partage de nom privé. Pour forcer le trait, on pourrait dire qu'il y a isomorphisme entre le calcul de processus et la combinatoire de la biologie moléculaire.

L'enjeu de ce domaine naissant, à savoir la modélisation concurrente des systèmes biologiques, est double. Il y a tout d'abord un immense champ applicatif potentiel pour les formalismes qui sauront proposer un niveau d'idéalisation satisfaisant pour le biologiste et allier lisibilité et facilité d'analyse. Il y a deuxièmement la perspective de renouveler l'étude des algèbres de processus. D'ores et déjà les travaux motivés par le premier enjeu mènent à l'exploration de nouveaux principes élémentaires de la communication, et tout particulièrement : la transcription formelle de la notion physique de contact (l'unique mode de synchronisation des processus cellulaires) [DL03b], la notion de groupe de calcul (pour représenter les réactants), la prise en compte de la réversibilité des interactions [DK03] et la nécessité de modéliser migration et transport biologiques et notamment les interactions de membranes et la perméabilité (modèles de bio-ambients et calcul de membranes de [RPS$^+$03, Car03]). On peut imaginer sans effort que le fait de se frotter à des systèmes computationnels dont les principes de fonctionnement sont loins de ceux de l'ingéniérie logicielle habituelle débouche à terme vers la conception de paradigmes de calcul non-conventionnels et pourquoi pas implantables dans les systèmes biologiques.[27]

Prenant en compte le premier impératif de lisibilité, Danos et Laneve ont construit des formalismes de représentation des évènements moléculaires de base. Ces formalismes se déclinent en langage de réécriture de multiensembles et langage de réécriture de graphes. Ces derniers permettent de représenter une partie significative des réseaux biologiques comme Chiaverini et Danos en ont donné la démonstration [CD03] en formalisant une compilation des principaux acteurs du cycle cellulaire des mammifères : le système obtenu rassemble 70 protéines et complexes et 700 réactions et se

[27] Un atelier international "Biology and Concurrency" satellite de la conférence internationale CONCUR'03, a fait le point sur cette interaction naissante entre concurrence et modélisation biologique en Septembre 2003.

prête bien à du requêtage qualitatif [CCD+03].[28]

Danos et Laneve donnent des compilations de ces formalismes de base dans des algèbres de processus intermédiaires qui elles-mêmes peuvent être plongées dans le π-calcul. Outre le fait que ces systèmes re-contextualisent les notions de concurrence dans un cadre biologique et notamment le concept traditionnel de bisimulation (qui permet de formuler la correction de ces traductions), les compilations données pour les formalismes de base sont des implantations canoniques de mécanismes de transactions. Elles sont basées respectivement sur deux principes fondamentaux en algorithmique distribuée : le consensus fondé sur les images locales où chaque agent est "conscient" de l'état des connexions qu'il a avec l'environnement [DL03a], et le consensus dynamique fondé sur l'exploration spontanée de voisinage par les agents. Ce dernier mécanisme est asynchrone et plus équilibriste du point de vue de la correction [DL03c] ; il donne une décomposition en interactions élémentaires binaires de toute réécriture de graphes et peut être raffiné en une forme d'évaluation symbolique de la cinétique de la réaction correspondante. Par ailleurs il permet d'apprécier (de donner en fait une borne supérieure sur) le coût que représenterait pour la cellule le fait de se passer de l'utilisation de complexes (assemblage protéiniques de faible énergie) pour ne se reposer que sur des activations (interactions avec complexation transitoire).

On peut comparer ces travaux à ceux de l'ingéniérie des circuits génétiques où le propos est de définir et d'implémenter dans les systèmes biologiques un jeu de composants synchrones élémentaires [HMC02]. Les décompositions proposées semblent une étape importante dans la compréhension de l'autre aspect de la question du calcul biologique, à savoir comment se servir, dans un monde où l'asynchronie est la règle, de synchronisations binaires de bas niveau comme de briques élémentaires pour construire des synchronisations arbitraires.

On attend aussi de ce type d'analyse des applications dans le domaine du calcul amorphe [NKC03] et en robotique distribuée [Kla02], et tout particulièrement des mécanismes puissants d'auto-assemblage. Tarissan a rendu explicite l'algorithme de consensus sous-jacent aux problèmes d'auto-assemblages par image locale posés par Klavins et a ainsi pu en donner une preuve rigoureuse fondée sur la bisimulation [Tar03]. On recueille ici le fruit d'une approche plus théorique qui passe par les algèbres de processus. Il semble d'ailleurs pertinent de monter encore en généralité et de rassembler les diverses "algèbres d'assemblage" proposées en un système de réécriture non-linéaire d'hypergraphes. D'autant que les techniques récentes développées dans [DL03b] résolvent le problème de l'auto-assemblage dans

[28] Une partie de ces travaux a été menée dans le cadre d'une Action de Recherche Concertée financée par l'INRIA sur la période 2002/2004 et en collaboration avec des chercheurs de l'Institut Pasteur.

un cadre considérablement plus général, débarassé d'hypothèse sur la préservation des composants élémentaires. Tarissan a entrepris de formuler une stratégie efficace de gestion des échecs transactionnels, et de donner une version spatialisée de l'algèbre de processus employée dans ces décompositions (on ne travaile pas directement en π-calcul pour ne pas se perdre dans la syntaxe, mais on fournit des compilations qui le prennent pour cible) qui permettent d'incorporer dans la preuve de correction de ces méthodes de décentralisation des éléments directement liés à la métrique ambiante : sphère de perception, distance d'interaction, perte de cohésion des assemblages.

Abordant un autre aspect des systèmes biologiques, Danos et Jean Krivine (DEA de Programmation) [DK03] ont récemment proposé une version de CCS qui exprime de manière simple la décomplexation et le changement d'état par l'introduction d'une distinction entre actions réversibles et irréversibles. C'est un exemple de modification apportée à un langage beaucoup étudié, motivé par une application biologique et qui a par ailleurs du potentiel dans d'autres champs applicatifs. Dans ce cas particulier, on pense à des systèmes de modélisation de transactions distribuées où la possibilité de retour-arrière offerte par la syntaxe permet de simplifier la description symbolique des algorithmes. Il se trouve que Krivine démontre également que la notion de réversibilité proposée caractérise l'équivalence par permutations en CCS et fournit l'équivalent de ce qu'on appelle l'étiquetage de Lévy en λ-calcul [Kri03a]. Ceci ouvre de très interessantes perspectives dans la théorie des algèbres de processus classiques. Ce travail est donc aussi un exemple qui illustre bien le renouvellement des formalismes de la concurrence auquel incite la modélisation de certains aspects des systèmes biologiques.

Références

[AC98] Roberto Amadio and Pierre-Louis Curien. *Domains and Lambda-calculi*. Cambridge University Press, 1998.

[ACCL91] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 4(1) :375–416, 1991.

[AG00] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236 :133–178, 2000.

[AM97] Samson Abramsky and Guy McCusker. Call-by-value games. In Morgens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic '97*, Lecture Notes in Computer Science, 1997.

[AM99] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *Logic in Computer Science 99*, pages

431–442, Trento, July 1999. IEEE, IEEE Computer Society Press.

[BA03] Anne-Gwenn Bosser and Francisco Alberti. L'expérience SCOL, un langage pour des applications internet multi-utilisateurs. In *Journées Francophones des Langages Applicatifs*. INRIA, 2003.

[Bar03] Sylvain Baro. *Conception et implémentation d'un système d'aide à la spécification et à la preuve de programmes ML.* Thèse de doctorat, Université Paris 6, July 2003.

[BB02] Stefano Berardi and Chantal Berline. Beta-eta complete models for system F. *Math. Struct. in Comput. Sci.*, 12 :823–874, 2002. Prépublication PPS//02/02//n°2 (pp).

[BB04] Stefano Berardi and Chantal Berline. Building continuous webbed models for system F. *Theor. Comput. Sci.*, 2004. Special issue MFPS '98. Prépublication PPS//02/02//n°3 (pp).

[BBLRD96] Zine-El-Abidine Benaissa, Daniel Briaud, Pierre Lescanne, and Jocelyne Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5) :699–722, 1996.

[BC82] Gérard Berry and Pierre-Louis Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20 :265–321, 1982.

[BC00] Mathilde Boehm and Emmanuel Chailloux. Étude de faisabilité de la traduction de ml vers c/c++. Rapport de fin d'étude pour le CEA, October 2000.

[BD02] Vincent Balat and Olivier Danvy. Memoization in type-directed partial evaluation. In *ACM SIGPLAN/SIGSOFT conference on Generative Programming and Component Engineering (GCSE/SAIG)*, number 2487 in Lecture Notes in Computer Science, Pittsburgh, USA, October 2002.

[BD03] Emmanuel Beffara and Vincent Danos. Disjunctive normal forms and local exceptions. In *International Conference on Functional Programming*, Uppsala, Sweden, August 2003.

[BDC99] Vincent Balat and Roberto Di Cosmo. A linear logical view of linear type isomorphisms. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *Lecture Notes in Computer Science*, pages 250–265. Springer, 1999.

[BDCF02] Vincent Balat, Roberto Di Cosmo, and Marcelo Fiore. Remarks on isomorphisms in typed lambda calculi with empty and sum types. In Gordon D. Plotkin, editor, *Proceedings of*

the Seventeenth Annual IEEE Symposium on Logic in Computer Science, Copenhagen, Denmark, July 2002. IEEE Computer Society Press.

[BDCF04] Vincent Balat, Roberto Di Cosmo, and Marcelo Fiore. Extensional normalisation and type-directed partial evaluation for typed lamda calculus with sums. In *31st Principles of Programming Languages*. ACM, 2004.

[BDM+02a] Anne Brygoo, Titou Durand, Pascal Manoury, Christian Queinnec, and Michèle Soria. Experiment around a training engine. In *IFIP 2002 - World Computer Congress*, August 2002.

[BDM+02b] Anne Brygoo, Titou Durand, Pascal Manoury, Christian Queinnec, and Michèle Soria. Un cédérom pour Scheme, chacun son entraîneur, un entraîneur pour tous. In *Colloque International sur les T.I.C.E. d'ingénieur et de l'industrie*, November 2002.

[BDP02] C. Baldi, Pierpaolo Degano, and Corrado Priami. Causal π-calculus for biochemical modeling. In *Proceedings of the AI*IA Workshop on BioInformatics 2002*, pages 69–72, 2002.

[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.

[Ber00] Chantal Berline. From computation to foundations via functions and application : the lambda-calculus and its webbed models. *Theor. Comput. Sci.*, 249 :81–161, 2000.

[BG97] Chantal Berline and Klaus Grue. A kappa-denotational semantics for map theory in ZFC+SI. *Theoretical Computer Science*, 179 :137–202, 1997.

[BKR00] Eduardo Bonelli, Delia Kesner, and Alejandro Ríos. A de Bruijn notation for higher-order rewriting. In Leo Bachmair, editor, *11th International Conference on Rewriting Techniques and Applications*, volume 1833 of *Lecture Notes in Computer Science*, pages 62–79. Springer-Verlag, July 2000.

[BKR01] Eduardo Bonelli, Delia Kesner, and Alejandro Ríos. From higher-order to first-order rewriting (extended abstract). In Aart Middeldorp, editor, *12th International Conference on*

Rewriting Techniques and Applications, volume 2051 of *Lecture Notes in Computer Science*, pages 47–62. Springer-Verlag, May 2001.

[BL85] Kim Bruce and Giuseppe Longo. Provable isomorphims and domain equations in models of typed languages. In *A.C.M. Symposium on Theory of Computing (STOC 85)*, 1985.

[BL03] Antonio Bucciarelli and Benjamin Leperchey. Hypergraphs and degrees of parallelism : a completeness result. Prépublication électronique PPS//03/10//n°24 (pp), Laboratoire Preuves, Programmes et Systèmes, October 2003. Submitted for publication.

[BLP03] Antonio Bucciarelli, Benjamin Leperchey, and Vincent Padovani. Relative definability and models of unary PCF. In *6th Int. Conf. on Typed Lambda Calculi and Applications*. Springer Verlag, 2003.

[BM02] Antonio Bucciarelli and Pasquale Malacaria. Relative definability of boolean functions via hypergraphs. *Theoretical Computer Science*, 278 :91–110, 2002.

[BM03a] Sylvain Baro and Pascal Manoury. Un système X. Raisonner formellement sur les programmes ML. In *Journées Francophones des Langages Applicatifs*. INRIA, January 2003. Prépublication PPS//02/12//n°12 (pp).

[BM03b] Sylvain Baro and François Maurel. The $q\nu$ and $q\nu K$ calculi : name capture and control — Extended Abstract. Prépublication électronique PPS//03/11//n°16 (pp), Laboratoire Preuves, Programmes et Systèmes, November 2003. Submitted for publication.

[Bon03] Eduardo Bonelli. A normalisation result for higher-order calculi with explicit substitutions. In *Foundations of Software Science and Computational Structures (FoSSACS)*, Lecture Notes in Computer Science. Springer Verlag, 2003.

[BPS03] Antonio Bucciarelli, Adolfo Piperno, and Ivano Salvo. Intersection types and lambda definability. *Mathematical Structures in Computer Science*, 13(1) :15–53, 2003.

[BS03] Antonio Bucciarelli and Antonino Salibra. The minimal graph model of lambda calculus. In *28th Int. Symp. on Mathematical Foundations of Computer Science*. Springer Verlag, 2003.

[Bur93] Albert Burroni. Higher dimensional word problems with applications to equational logic. *Theoretical Computer Science*, 115 :43–62, 1993.

[Bur03a] Albert Burroni. Automates et grammaires en dimensions supérieures. Submitted, 2003.

[Bur03b] Albert Burroni. Une nouvelle approche des orientaux de Street : application à l'homotopie dirigée. Submitted, 2003.

[Car03] Luca Cardelli. Brane calculi. To Appear in the ENTCS Proceedings of Bio-Concur, 2003.

[CCD+03] Nathalie Chabrier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and Querying biological networks. Submitted to Theoretical Computer Science, 2003.

[CCE04] Clément Capel, Emmanuel Chailloux, and Jean-Marc Eber. Applications du toplevel embarqué d'Objective Caml. Submitted to JFLA'04, 2004.

[CCF94] Robert Cartwright, Pierre-Louis Curien, and Matthias Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 111(2) :297–401, 1994.

[CD03] Marc Chiaverini and Vincent Danos. A core modeling language for the working molecular biologist. Communication to the International Workshop on Computational Methods in Systems Biology, 2003.

[CF92] Robert Cartwright and Matthias Felleisen. Observable sequentiality and full abstraction. In *Proc. ACM Principles of Prog. Lang.*, 1992.

[CF03a] Emmanuel Chailloux and Christian Foisy. A portable implementation for Objective Caml Flight. In *Second Workshop on High-Level Parallel Programming and Applications*, June 2003.

[CF03b] Emmanuel Chailloux and Christian Foisy. A portable implementation for Objective Caml Flight. *Parallel Processing Lettersi*, 13(3), 2003. Special issue on High Level Parallel Programming and Applications.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *Proceedings of the International Conference on Functional Programming*, September 2000.

[CH04] Emmanuel Chailloux and Grégoire Henry. O'Jacaré : une interface objet entre Objective Caml et Java. Submitted to LMO'04, 2004.

[Cha02a] Emmanuel Chailloux. Cast Objet pour la construction de hiérarchies de classes en Objective Caml. In INRIA, editor, *Actes des Journées Francophones des Langages Applicatifs*, Anglet, January 2002.

[Cha02b] Emmanuel Chailloux. Dynamic object typing in objective caml. In *International LISP Conference 2002*, San Francisco, October 2002.

[CHL96] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Weak and strong confluent calculi of explicit substitutions. *JACM*, 43(2), 1996.

[Chr00] Juliusz Chroboczek. Game Semantics and subtyping. In *Proceedings of the fifteenth annual IEEE symposium on Logic in Computer Science*, Santa Barbara, California, June 2000.

[Chr01a] Juliusz Chroboczek. *Game Semantics and subtyping.* PhD thesis, University of Edinburgh, 2001.

[Chr01b] Juliusz Chroboczek. Subtyping recursive games. In *Proceedings of the Fifth International Conference on Typed Lambda Calculi and Applications (TLCA'01)*, Kraków, Poland, May 2001.

[Chr02] Juliusz Chroboczek. Lazy errors in an eager calculus. Unpublished draft, 2002.

[Ciz03] Gisèle Cizault. *IPv6, théorie et pratique.* O'Reilly, 2003. 3ème édition.

[CK04] Serenella Cerrito and Delia Kesner. Pattern matching as cut elimination. *Theoretical Computer Science*, 2004.

[CMP00] Emmanuel Chailloux, Pascal Manoury, and Bruno Pagano. *Développement d'applications avec Objective Caml.* O'Reilly, Paris, April 2000. Version en ligne : `http://www.oreilly.fr`, english on-line version : `http://caml.inria.fr/oreilly-book`.

[CMP04] Emmanuel Chailloux, Raphaël Montelatici, and Bruno Pagano. CamIL : un compilateur Objective Caml vers .NET. Submitted to RIVF '04, 2004.

[Cou99] Guy Cousineau. Interfaces visuelles. Notes de Cours de DESS, Université Paris 7 (160 pages), 1999.

[Cou00] Guy Cousineau. Characterization of some periodic patterns. In preparation, May 2000.

[Cou02] Guy Cousineau. Tilings as a programming exercise. *Theor. Comput. Sci.*, 281 :207–217, 2002.

[Cur] Pierre-Louis Curien. Abstract Böhm trees II : concrete notation and examples. In preparation.

[Cur93] Pierre-Louis Curien. On the symmetry of sequentiality. In *Proc. Mathematical Foundations of Programming Semantics 93*, pages 122–130. Springer Lect. Notes in Comp. Sci. 802, 1993.

[Cur98] Pierre-Louis Curien. Abstract Böhm trees. *Mathematical Structures in Computer Science*, 8(6) :559–591, 1998.

[Cur03] Pierre-Louis Curien. Sequential algorithms as bistable maps. Submitted, 2003.

[CVDCW03] François Clément, Arnaud Vodicka, Roberto Di Cosmo, and Pierre Weis. Couplage de codes numériques, parallélisme et langages de haut niveau. RR 4825, INRIA, 2003. `http://www.inria.fr/rrrt/rr-4825.html`.

[Dan03] Olivier Danvy, editor. *Special issue on Krivine's Abstract Machine*. Higher-Order and Symbolic Computation. Kluwer, 2003. To appear.

[DC03] Roberto Di Cosmo. Legal tools to protect software : Choosing the right one. *Upgrade*, 4(3) :21–23, June 2003. Available as `http://www.upgrade-cepis.org/issues/2003/3/up4-3DiCosmo.pdf`.

[DCK94] Roberto Di Cosmo and Delia Kesner. Simulating expansions without expansions. *Mathematical Structures in Computer Science*, 4 :1–48, 1994.

[DCK95] Roberto Di Cosmo and Delia Kesner. Rewriting with extensional polymorphic λ-calculus. In Hans Kleine Büning, editor, *Proceedings of the 9th Annual Conference of the European Association for Computer Science Logic*, volume 1092 of *Lecture Notes in Computer Science*, pages 215–232. Springer-Verlag, September 1995.

[DCK96a] Roberto Di Cosmo and Delia Kesner. Combining algebraic rewriting, extensional lambda calculi and fixpoints. *Theoretical Computer Science*, 169(2) :201–220, 1996.

[DCK96b] Roberto Di Cosmo and Delia Kesner. Strong normalization of explicit substitutions via cut elimination in proof nets (extended abstract). In *Twelfth Annual IEEE Symposium on Logic in Computer Science (LICS)*, 1996.

[DCKP03] Roberto Di Cosmo, Delia Kesner, and Emmanuel Polonovski. Proof nets and explicit substitutions. *Mathematical Structures in Computer Science*, 13(3) :409–450, June 2003.

[DCL00] Roberto Di Cosmo and Jean-Vincent Loddo. Playing logic programs with the alpha-beta algorithm. In Michel Parigot and Andrei Voronkov, editors, *LPAR'00*, volume 1955 of *Lecture Notes in Computer Science*, pages 207–224, 2000.

[DCP03] Roberto Di Cosmo and Susanna Pelagatti. A calculus for dense array distributions. *Second International Workshop on High-Level Parallel Programming and Applications*, 2003.

[DCPR03] Roberto Di Cosmo, François Pottier, and Didier Rémy. Subtyping recursive types modulo associative commutative products. Submitted, 2003.

[DCS03] Roberto Di Cosmo and Sergei Soloviev, editors. *Special Issue on Type Isomorphisms*, Mathematical Structures in Computer Science. Cambridge University Press, 2003. In preparation.

[DD02] Vincent Danos and Josée Desharnais. Une note sur les chaînes de Markov étiquetées. Manuscript, 2002.

[DD03a] Vincent Danos and Josée Desharnais. A fixpoint logic for labeled Markov Processes. In Zoltan Esik and Igor Walukiewicz, editors, *Proceedings of an international Workshop FICS'03 (Fixed Points in Computer Science)*, Warsaw, 2003.

[DD03b] Vincent Danos and Josée Desharnais. Labeled Markov Processes : Stronger and faster approximations. In *Proceedings of the 18^{th} Symposium on Logic in Computer Science*, Ottawa, 2003. IEEE Computer Society Press.

[DDP03] Vincent Danos, Josée Desharnais, and Prakash Panangaden. Conditional expectations and the approximation of labeled Markov Processes. In *Proceedings of CONCUR'03, Marseille, France*, volume 2761 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[Deh04] Patrick Dehornoy. Complete positive group presentations. To appear in *Journal of Algebra*, 2004.

[DG01] René David and Bruno Guillaume. A lambda-calculus with explicit substitution and explicit weakening. *Mathematical Structures in Computer Science*, 11 :169–206, 2001.

[DG03] Vincent Danos and Deepak Garg. Name-dependent typing for a higher-order communication language. In Preparation, 2003.

[DH00] Vincent Danos and Russell Harmer. Probabilistic game semantics (extended abstract). In *Proceedings, Fifteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 2000.

[DH01] Vincent Danos and Russell Harmer. The anatomy of innocence. In Laurent Fribourg, editor, *Proceedings, Tenth Annual Conference of the European Association for Computer Science Logic*, number 2142 in Lecture Notes in Computer Science, pages 188–202. Springer Verlag, 2001.

[DH02] Vincent Danos and Russell Harmer. Probabilistic game semantics. *ACM Transactions on Computational Logic*, 3(3) :359–382, 2002.

[DJS97] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic : Linear logic. *Journal of Symbolic Logic*, 62(3) :755–807, September 1997.

[DK00] Vincent Danos and Jean-Louis Krivine. Disjunctive tautologies and synchronisation schemes. In Peter Clote and Helmut Schwichtenberg, editors, *Proceedings of CSL'00*, volume 1862 of *Lecture Notes in Computer Science*, pages 292–301, Fischbachau, 2000. Springer-Verlag.

[DK03] Vincent Danos and Jean Krivine. Formal molecular biology done in CCS. In *Proceedings of BIO-CONCUR'03, Marseille, France*, Electronic Notes in Theoretical Computer Science. Elsevier, 2003.

[DL03a] Vincent Danos and Cosimo Laneve. Core formal molecular biology. In *Proceedings of the 12th European Symposium on Programming (ESOP'03, Warsaw, Poland)*, volume 2618 of *Lecture Notes in Computer Science*, pages 302–318. Springer-Verlag, April 2003.

[DL03b] Vincent Danos and Cosimo Laneve. Formal molecular biology. Submitted, September 2003.

[DL03c] Vincent Danos and Cosimo Laneve. Graphs for formal molecular biology. In *Proceedings of the First International Workshop on Computational Methods in Systems Biology (CMSB'03, Rovereto, Italy)*, volume 2602 of *Lecture Notes in Computer Science*, pages 34–46. Springer-Verlag, February 2003.

[DM02] David Delahaye and Micaela Mayero. Dealing with algebraic expressions over a (commutative) field in coq using maple. Submitted to Journal of Symbolic Computation : special issue on the integration of automated reasoning and computer algebra systems, 2002.

[dPS92] Ugo de'Liguoro, Adolfo Piperno, and Richard Statman. Retracts in simply typed lambda-beta-eta–calculus. In IEEE Computer Society Press, editor, *Proceeding of the 7th annual IEEE symposium on Logic in Computer Science (LICS92)*, 1992.

[DRR03] René David, Christophe Raffali, and Paul Rozière. Quelques éléments pour débuter avec le vérificateur de preuves PhoX. Une introduction rapide à PhoX, 2003.

[Dug01] Dominic Duggan. Higher-order substitutions. *Information and Computation*, 164(1) :1–53, 2001.

[EDI02] Rapport d'avancement du projet EDICA, livraison du lot 1, 2002. Laboratoire PPS, Laboratoire LIP6, Cryo-networks. Réseau National des Technologies Logicielles.

[EDI03] Rapport d'avancement du projet EDICA, livraison du lot 2, 2003. Laboratoire PPS, Laboratoire LIP6, Virtools. Réseau National des Technologies Logicielles.

[Ehr97] Thomas Ehrhard. A relative definability result for strongly stable functions and some corollaries. *Information and Computation*, 1997.

[FK03] Julien Forest and Delia Kesner. Expression reduction systems with patterns. In Robert Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science. Springer-Verlag, June 2003.

[Gaua] Philippe Gaucher. A Convenient Category for The Homotopy Theory of Concurrency. arXiv :math.AT/0201252.

[Gaub] Philippe Gaucher. A long exact sequence for the branching homology. arXiv :math.AT/0305169.

[Gauc] Philippe Gaucher. Homotopy branching space and weak dihomotopy. arXiv :math.AT/0304112.

[Gau00a] Philippe Gaucher. From concurrency to algebraic topology. In *Electronic Notes in Theoretical Computer Science*, volume 39, 2000.

[Gau00b] Philippe Gaucher. Homotopy invariants of higher dimensional categories and concurrency in computer science. *Math. Structures Comput. Sci.*, 10(4) :481–524, 2000.

[Gau01] Philippe Gaucher. Combinatorics of branchings in higher dimensional automata. *Theory and Applications of Categories*, 8(12) :324–376, 2001.

[Gau02a] Philippe Gaucher. About the globular homology of higher dimensional automata. *Cahiers Topologie Géom. Différentielle Catég.*, 43(2) :107–156, 2002.

[Gau02b] Philippe Gaucher. Investigating the algebraic structure of dihomotopy types. *Electronic Notes in Computer Science*, 52(2), 2002.

[Gau03a] Philippe Gaucher. The branching nerve of hda and the Kan condition. *Theory and Applications of Categories*, 11(3) :75–106, 2003.

[Gau03b] Philippe Gaucher. Concurrent process up to homotopy (I). *C. R. Acad. Sci. Paris Ser. I Math.*, 336(7) :593–596, 2003. French.

[Gau03c] Philippe Gaucher. Concurrent process up to homotopy (II). *C. R. Acad. Sci. Paris Ser. I Math.*, 336(8) :647–650, 2003. French.

[GG03] Philippe Gaucher and Eric Goubault. Topological deformation of higher dimensional automata. *Homology, Homotopy and Applications*, 5(2) :39–82, 2003.

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

[Gir01] Jean-Yves Girard. Locus solum : from the rules of logic to the logic of rules. *Math. Structures Comput. Sci.*, 11(3) :301–506, 2001.

[Gru92] Klaus Grue. Map theory. *Theor. Comput. Sci.*, 102 :1–133, 1992.

[GT01] Ronojoy Ghosh and Claire J. Tomlin. Lateral inhibition through delta-notch signaling : A piecewise affine hybrid model ? In *Proceedings of HSCC 2001*, volume 2034 of *LNCS*, pages 232–246. Springer, 2001.

[Har99] Russell Harmer. *Games and full abstraction for nondeterministic languages*. PhD thesis, University of London, 1999.

[HL89] Thérèse Hardin and Jean-Jacques Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, Izu (Japan), 1989.

[HM] Russell Harmer and Guy McCusker. Full abstraction for finite nondeterminism. Submitted for publication. Fully revised version of [HM99].

[HM99] Russell Harmer and Guy McCusker. A fully abstract game semantics for finite nondeterminism. In *Proceedings, Fourteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1999.

[HMC02] Jeff Hasty, David McMillen, and James J. Collins. Engineered gene circuits. *Nature*, 420 :224–230, November 2002.

[HMP98] Thérèse Hardin, Luc Maranget, and Bruno Pagano. Functional runtime systems within the lambda-sigma calculus. *Journal of Functional Programming*, 8(2) :131–176, 1998.

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

[HT98] R. Hofestädt and S. Thelen. Quantitative modeling of biochemical networks. *In Silico Biology*, 1, 1998. `http://www.bioinfo.de/isb/1998/`.

[ITR+01] Trey Ideker, Vesteinn Thorsson, Jeffrey A. Ranish, Rowan Christmas, Jeremy Buhler, Jimmy K. Eng, Roger Bumgarner, David R. Goodlett, Ruedi Aebersold, and Leroy Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic pathway. *Science*, 292 :929–934, May 2001.

[Joi02] Jean-Baptiste Joinet. Proofs, reasoning and the metamorphosis of logic. Submitted, 2002.

[Joi03] Jean-Baptiste Joinet. Le temps logique. In *Logique mathématique, Informatique et Philosophie*. Presses Universitaires de la Sorbonne, 2003. Actes du colloque éponyme.

[Jol00a] Thierry Joly. *Codages, séparabilité et représentation de fonctions en λ-calcul simplement typé et dans d'autres systèmes de types*. Thèse de Doctorat, Université Paris VII, January 2000.

[Jol00b] Thierry Joly. Non finitely generated types & λ-terms combinatoric representation cost. *C.R.A.S. of Paris, Série I*, 331 :581–586, 2000.

[Jol01a] Thierry Joly. Constant time parallel computations in λ-calculus. *Theoretical Computer Science*, 266 :975–985, 2001.

[Jol01b] Thierry Joly. The finitely generated types of the λ-calculus. In *Typed Lambda Calculi and Applications 2001*, volume 2044 of *LNCS*, pages 240–252, 2001.

[Jol03a] Thierry Joly. About λ-definability and combinatoric generation of types. (Résultats exposés par H. Barendregt aux "2002 Clifford Lectures"), 2003.

[Jol03b] Thierry Joly. Encoding of the halting problem into the monster type & applications. In *Typed Lambda Calculi and Applications 2003*, volume 2701 of *LNCS*, pages 153–166, 2003.

[Jol03c] Thierry Joly. The ξ-rule of the untyped λ-calculus is not finitely axiomatizable. Soumis au Journal of Functional Programming, 2003.

[JSTdF02] Jean-Baptiste Joinet, Harold Schellinx, and Lorenzo Tortora de Falco. Strong normalization and Church-Rosser property for free-style LK^{tq} : linear decorations and simulation of normalization. *Journal of Symbolic Logic*, 67(1) :162–196, 2002.

[KCH01] Naaman Kam, Irun R. Cohen, and David Harel. The immune system as a reactive system : modeling T-cell activation with statecharts. In *Proceedings of the IEEE Symposium on Human-centric Computing*, pages 15–22, September 2001.

[KEG] Kyoto Encyclopedia of Genes and Genomes, Bioformatics Center, Institute for Chemical Research, Kyoto University. On-line data-base, `http://www.genome.ad.jp/kegg`.

[Kes96] Delia Kesner. Confluence properties of extensional and non-extensional λ-calculi with explicit substitutions. In Harald Ganzinger, editor, *Seventh International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture*

Notes in Computer Science, pages 184–199. Springer Verlag, July 1996.

[Kes00] Delia Kesner. Confluence of extensional and non-extensional lambda-calculi with explicit substitutions. *Theoretical Computer Science*, 238(1-2) :183–220, 2000.

[Kha90] Zurab Khasidashvili. Expression reduction systems. In *Proceedings of IN Vekua Institute of Applied Mathematics*, volume 36, Tbilisi, 1990.

[Kla02] Eric Klavins. Automatic synthesis of controllers for distributed assembly and formation forming. In *Proceedings of the 2002 International Conference on Robotics and Automation*, 2002.

[KR95] Fairouz Kamareddine and Alejandro Ríos. A λ-calculus à la de Bruijn with explicit substitutions. In Doaitse Swierstra and Manuel Hermenegildo, editors, *Proceedings of the 7th International Symposium on Programming Languages Implementation and Logic Programming (PLILP)*, volume 982 of *Lecture Notes in Computer Science*, pages 45–62. Springer Verlag, September 1995.

[KR97] Fairouz Kamareddine and Alejandro Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4) :395–420, 1997.

[Kri01] Jean-Louis Krivine. Typed lambda-calculus in classical Zermelo-Fraenkel set theory. *Archive for Mathematical Logic*, 40(3) :189–205, 2001.

[Kri03a] Jean Krivine. Algèbre de processus réversibles. Mémoire de DEA, disponible au `http://pauillac.inria.fr/~krivine/`, September 2003.

[Kri03b] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 2003. To appear.

[Kri03c] Jean-Louis Krivine. Dependent choice, 'quote' and the clock. *Theoretical Computer Science*, 2003. To appear.

[Lai02] James Laird. Bistability and bisequentiality. Draft, 2002.

[Lai03] James Laird. A fully bidomain model of unary PCF. In *Typed Lambda Calculus and Applications '03*. Springer Verlag, 2003.

[Lam92] François Lamarche. Sequentiality, games and linear logic. *Manuscript*, 1992.

[Lau99] Olivier Laurent. Polarized proof-nets : proof-nets for LC (extended abstract). In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications '99*, volume 1581 of *Lecture Notes in Computer Science*, pages 213–227. Springer, April 1999.

[Lau02a] Olivier Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, March 2002.

[Lau02b] Olivier Laurent. Polarized games (extended abstract). In *Proceedings of the seventeenth annual symposium on Logic In Computer Science*, pages 265–274, Copenhagen, July 2002. IEEE, IEEE Computer Society Press.

[Lau03a] Olivier Laurent. Classical isomorphisms of types. Prépublication électronique PPS//03/04//n°18 (pp), Laboratoire Preuves, Programmes et Systèmes, March 2003. Submitted to a special issue of *Mathematical Structures in Computer Science* [DCS03].

[Lau03b] Olivier Laurent. Polarized games. Submitted to *Annals of Pure and Applied Logic*, February 2003.

[Lau03c] Olivier Laurent. Polarized proof-nets and $\lambda\mu$-calculus. *Theoretical Computer Science*, 290(1) :161–188, January 2003.

[Lau03d] Olivier Laurent. Syntax vs. semantics : a polarized approach. Prépublication électronique PPS//03/04//n°17 (pp), Laboratoire Preuves, Programmes et Systèmes, March 2003. Submitted to a special issue of *Theoretical Computer Science* on games.

[Lep03] Benjamin Leperchey. SR functionals and "quote". En préparation, 2003.

[Lev] Paul Blain Levy. From typed call-by-value and typed call-by-name to call-by-push-value. Submitted.

[Lev99] Paul Blain Levy. Call-by-push-value : A subsuming paradigm (extended abstract). In Jean-Yves Girard, editor, *Typed Lambda-Calculi and Applications '99*, volume 1581 of *Lecture Notes in Computer Science*, pages 228–242. Springer, April 1999.

[Lev01] Paul Blain Levy. *Call-by-push-value*. PhD thesis, Queen Mary, University of London, March 2001.

[Lev02] Paul Blain Levy. Possible world semantics for general storage in call-by-value. In J. Bradfield, editor, *Proceedings of 16th Annual Conference in Computer Science Logic, Edinburgh, 2002*, volume 2471 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 2002.

[Lev03] Paul Blain Levy. Adjunction models for call-by-push-value with stacks. In *Proceedings of 9th Conference on Category Theory and Computer Science, Ottawa, 2002*, volume 69 of *Electronic Notes in Theoretical Computer Science*, February 2003.

[Lod02] Jean-Vincent Loddo. *Généralisation des Jeux Combinatoires et Applications aux Langages Logiques*. Thèse de doctorat, Université Paris 7, Paris, December 2002.

[Lod03] Jean-Vincent Loddo. A generalization of combinatorial game theory and its application to constrained logic programming. *Submitted to Theoretical Computer Science*, 2003.

[Lon99] John Longley. When is a functional program not a functional program ? In *Int. Conf. on Functional Programming*. ACM Press, 1999.

[LQTdF00] Olivier Laurent, Myriam Quatrini, and Lorenzo Tortora de Falco. Polarized and focalized linear and classical proofs. Prépublication 24, Institut de Mathématiques de Luminy, Marseille, France, September 2000. To appear in *Annals of Pure and Applied Logic*.

[LR03] Olivier Laurent and Laurent Regnier. About translations of classical logic into polarized linear logic. In *Proceedings of the eighteenth annual symposium on Logic In Computer Science*. IEEE, IEEE Computer Society Press, June 2003.

[LTdF01] Olivier Laurent and Lorenzo Tortora de Falco. Slicing polarized additive normalization. Quaderno 12, Istituto per le Applicazioni del Calcolo (CNR), Roma, Italia, March 2001. To appear in the TMR LINEAR book *Linear Logic in Computer Science*.

[LTP04] Paul Blain Levy, Hayo Thielecke, and John Power. Modelling environments in call-by-value programming languages. *Information and Computation*, 2004. To appear.

[Mau03a] François Maurel. Non Deterministic Light Logics and NP-Time. In *Typed Lambda Calculus and Applications*, 2003.

[Mau03b] François Maurel. Ocaml-templates, gérénation de code à partir de types. Submitted, 2003.

[MDNM00] Hiroshi Matsuno, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Hybrid Petri Net representation of gene regulatory networks. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 338–349, 2000.

[Mel98] Paul-André Melliès. Axiomatic rewriting 4 : a stability theorem in rewriting theory. In *Logic in Computer Science '98*. IEEE, IEEE Computer Society Press, July 1998.

[Mel00] Paul-André Melliès. Axiomatic rewriting 2 : The lambda-sigma calculus enjoys finite normalisation cones. *Journal of Logic and Computation*, 10(3) :461–487, 2000.

[Mel02a] Paul-André Melliès. Axiomatic rewriting 1 : a diagrammatic standardization theorem. Prépublication électronique PPS//02/12//n°13 (pp), Laboratoire Preuves, Programmes et Systèmes, December 2002. Submitted to *Information and Computation*.

[Mel02b] Paul-André Melliès. Axiomatic rewriting 6 : Residual theory revisited. In Sophie Tison, editor, *Rewriting Techniques and Applications '02*, volume 2378 of *Lecture Notes in Computer Science*, pages 24–50. Springer Verlag, July 2002.

[Mel02c] Paul-André Melliès. Comparing hierarchies of types in models of linear logic. Prépublication électronique PPS//02/12//n°14 (pp), Laboratoire Preuves, Programmes et Systèmes, December 2002. Submitted to *Information and Computation*.

[Mel02d] Paul-André Melliès. Double categories : a modular model of multiplicative linear logic. *Mathematical Structure in Computer Science*, 12 :449–479, 2002.

[Mel02e] Paul-André Melliès. A topological correctness criterion for non-commutative logic. Prépublication électronique PPS//02/12//n°15 (pp), Laboratoire Preuves, Programmes, Systèmes, December 2002. To appear in the TMR LINEAR book *Linear Logic in Computer Science.*

[Mel03a] Paul-André Melliès. Asynchronous games 1 : an orbital formulation of arena games. Submitted to *Mathematical Structure in Computer Science*, 2003.

[Mel03b] Paul-André Melliès. Asynchronous games 2 : innocent strategies are positional. En préparation, 2003.

[Mel03c] Paul-André Melliès. Categorical models of linear logic revisited. Prépublication électronique PPS//03/09//n°22 (pp), Laboratoire Preuves, Programmes et Systèmes, March 2003. Submitted to *Theoretical Computer Science*.

[Mel03d] Paul-André Melliès. Sequential algorithms and strongly stable functions. Prépublication électronique PPS//03/09//n°23 (pp), Laboratoire Preuves, Programmes et Systèmes, April 2003. Submitted to a special issue of *Theoretical Computer Science* on games.

[Mel03e] Paul-André Melliès. Where Böhm trees collapse as sequential algorithms. En préparation, 2003.

[Miq01] Alexandre Miquel. *Le calcul des constructions implicite : syntaxe et sémantique*. Thèse de doctorat, Université Paris 7, 2001.

[Miq03] Alexandre Miquel. A strongly normalising Curry-Howard correspondence for IZF. In *Proceedings of Computer Science Logic (CSL'03)*, 2003.

[Mon01] Raphaël Montelatici. Présentation axiomatique de théorèmes de complétude forte en sémantique des jeux et en logique classique. Mémoire de D.E.A de programmation, Univ. Paris 7, September 2001.

[Mon03] Raphaël Montelatici. Polarized Proof Nets with Cycles and Fixpoints Semantics. In *Typed Lambda Calculus and Applications*, 2003.

[MPS86] David MacQueen, Gordon Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71(1-2) :95–130, 1986.

[MS03] Paul-André Melliès and Peter Selinger. On polarized games and polarized categories. En préparation, 2003.

[Muñ97] César Muñoz. A left-linear variant of $\lambda\sigma$. In Michael Hanus, Jan Heering, and Karl Meinke, editors, *Proceedings of the 6th International Conference on Algebraic and Logic Programming (ALP'97)*, volume 1298 of *Lecture Notes in Computer Science*. Springer-Verlag, September 1997.

[MV03] Paul-André Melliès and Jérôme Vouillon. A realizability model of recursive types. En préparation, 2003.

[MW03] Alexandre Miquel and Benjamin Werner. The not so simple proof-irrelevant model of CC. In Herman Geuvers and Freek Wiedijk, editors, *Types for Proofs and Programs (TYPES 2002)*, volume 2646 of *Lecture Notes in Computer Science*, pages 240–258. Springer-Verlag, 2003.

[Nic94] Hanno Nickau. Hereditarily sequential functionals. In *Proceedings of the Symposium on Logical Foundations of Computer Science : Logic at St. Petersburg*, 1994.

[NKC03] Radhika Nagpal, Attila Kondacs, and Catherine Chang. Programming methodology for biologically-inspired self-assembling systems. In *AAAI Spring Symposium on Computational Synthesis : From Basic Building Blocks to High Level Functionality*, March 2003.

[Pad96] Vincent Padovani. Decidability of all minimal models. In Springer, editor, *Proceedings of Types for Proofs and Programs - Torino 1995*, volume 1158 of *Lecture Notes in Computer Science*, 1996.

[Pad01] Vincent Padovani. Retracts in simple types. In Springer, editor, *Proceedings of Typed Lambda Calculi and Applica-*

tions (TLCA'01), volume 2004 of *Lecture Notes in Computer Science*, 2001.

[Par00a] Michel Parigot. On the computational interpretation of negation. In *Computer Science Logic (CSL'00)*, volume 1862 of *Lectures Notes in Computer science*, 2000.

[Par00b] Michel Parigot. Strong normalisation of second order symmetric λ-calculus. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'00)*, volume 1974 of *Lectures Notes in Computer science*, 2000.

[PRSS01] Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80 :25–31, 2001.

[PV03] Benjamin Pierce and Jérôme Vouillon. How to specify a file synchronizer. Submitted, 2003.

[QC02] Christian Queinnec and Emmanuel Chailloux. Une expérience de notation de masse. In *TICE 2002 - Technologies de l'Information et de la Communication dans les Enseignements d'Ingénieurs et dans l'industrie - Conférences ateliers*, November 2002.

[Rif02] Jean-Marie Rifflet. *Programming under ChorusOS*. En ligne sur Internet, 2002. Publication par SunPress annulée suite au lachâge par SUN de Chorus.

[Roc02] Jérôme Rocheteau. $\lambda\mu$-calcul et dualité : appel par nom et appel par valeur. Prépublication électronique PPS//02/10//n°8 (mn), Laboratoire Preuves, Programmes et Systèmes, September 2002. Mémoire du D.E.A. de Logique Mathématique et Fondements de l'Informatique.

[RPS+03] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients : An abstraction for biological compartments. *Theoretical Computer Science*, 2003. To Appear.

[RR02] Christophe Raffalli and Paul Rozière. The proof checker documentation. Manuel de l'assistant de preuve PhoX, 2002.

[RS02] Aviv Regev and Ehud Shapiro. Cells as computation. *Nature*, 419, September 2002.

[RSS01] Aviv Regev, William Silverman, and Ehud Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470, Singapore, 2001. World Scientific Press.

[RY03] Jean-Marie Rifflet and Jean-Baptiste Yunès. *UNIX : Programmation et Commnunication*. Dunod, August 2003. En ligne sur Internet.

[San01] Davide Sangiorgi. Termination of processes. Draft, September 2001.

[SEJGM02] Birgit Schoeberl, Claudia Eichler-Jonsson, Ernst-Dieter Gilles, and Gertraud Müller. Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology*, 20 :370–375, 2002.

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

[Tar03] Fabien Tarissan. Self-assembly and formal molecular biology. Mémoire de D.E.A de programmation, Université Paris VII, September 2003.

[Val01] Thierry Vallée. *"Map Theory" et antifondation*. Thèse de doctorat, Université Paris 7, December 2001. Parue dans ENTCS vol.79 (2003).

[Val03] Thierry Vallée. Map Theory : from well foundation to antifoundation. In *Proceedings of the 6th Workshop on Domain Theory*, 2003.

[VM04] Jérôme Vouillon and Paul-André Melliès. Semantic types : A fresh look at the ideal model for types. In *Proceedings of the 31st ACM Conference on Principles of Programming Languages*, Venise, Italie, January 2004.