INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet CALLIGRAMME

Logique Linéaire, Réseaux de Démonstration et Grammaires Catégorielles

Nancy

THÈME 2A

Table des matières

1 Composition de l'équipe **2**

2 Présentation et objectifs généraux **2**

3 Fondements scientifiques **3**
3.1 La logique linéaire . . . 3
3.2 Calcul des séquents, réseaux de démonstration et d'interaction . . . 5
3.3 Logique linéaire et parallélisme . . . 7
3.4 Logique linéaire et grammaires formelles . . . 9
3.5 Lambda calculs typés, sémantique, complexité . . . 11

4 Domaines d'applications **14**
4.1 Les grammaires catégorielles dans la linguistique informatique . . . 14
4.2 Analyse syntaxique . . . 14
4.3 Génération automatique de textes . . . 15

5 Résultats nouveaux **15**
5.1 Calcul des séquents, réseaux de démonstration et d'interaction . . . 15
5.2 Logique linéaire et grammaires formelles . . . 15
5.3 Lambda calculs typés, sémantique, complexité . . . 19

6 Contrats industriels (nationaux, européens et internationaux) **22**
6.1 Collaboration avec Xerox . . . 22

7 Actions régionales, nationales et internationales **22**
7.1 Actions nationales . . . 22
7.2 Actions européennes . . . 23
7.3 Actions internationales . . . 23
7.4 Visites et invitations de chercheurs . . . 23

8 Diffusion de résultats **24**
8.1 Animation de la Communauté scientifique . . . 24
8.2 Enseignement . . . 24
8.3 Jurys de thèse . . . 24
8.4 Participation à des colloques, séminaires, invitations . . . 25

9 Bibliographie **26**

CALLIGRAMME est un projet du LORIA (UMR 7503) commun au CNRS, à l'INRIA, à l'université Henri POINCARÉ *Nancy 1, à l'Université Nancy 2 et à l'Institut National Polytechnique de Lorraine.*

1 Composition de l'équipe

Responsable scientifique

Philippe de Groote [CR Inria]

Responsable permanent

François Lamarche [DR Inria]

Assistantes de projet

Anne Loth [CDD, Fevrier à Octobre]

Geneviève Pierrelée [jusqu'au 31 Janvier, et depuis 15 Octobre]

Personnel Universitaire

Adam Cichon [Professeur, Université Henri Poincaré]

Alain Lecomte [Professeur, Université de Grenoble 2]

Jean-Yves Marion [Maître de conférences, Université Nancy 2, détaché à l'INRIA depuis Octobre 99]

Guy Perrier [Maître de conférences, Université Nancy 2]

Chercheurs doctorants

Guillaume Bonfante [Institut National Polytechnique de Lorraine, ATER Université Henri Poincaré]

Catherine Pilière [Université Henri Poincaré,ATER IUT de Metz (depuis le 1er septembre 1999)]

Sylvain Pogodalla [boursier CIFRE, Institut National Polytechnique de Lorraine]

Jérôme Besombes [allocataire MESR, Université Henri Poincaré, Moniteur à Université de Metz (depuis le 1er septembre 1999)]

2 Présentation et objectifs généraux

La création officielle du projet Calligramme s'est effectuée en mai 1996. Le thème de recherche de base du projet est la logique linéaire : sa syntaxe, sa sémantique, ses rapports avec les formalismes constructifs traditionnels, mais plus spécifiquement l'étude des réseaux de démonstration et de leurs proches parents, les réseaux d'interaction. Les applications de ces recherches sont orientées vers le calcul parallèle (calculs de processus et langages de programmation spécialisés), et surtout vers la linguistique : une grande part de nos recherches porte

sur le développement du formalisme des grammaires catégorielles et l'étude de leurs relations avec la sémantique, ceci dans le but de construire des analyseurs syntaxiques et sémantiques plus performants et plus faciles à intégrer à des systèmes complexes.

Mots clés : logique linéaire, calcul des séquents, réseau de démonstration, réseau d'interaction, grammaire catégorielle, analyse syntaxique des langues naturelles, sémantique des langues naturelles, calcul parallèle, calcul de processus, lambda-calcul, théorie des types, sémantique dénotationnelle, sémantique des jeux, complexité.

3 Fondements scientifiques

3.1 La logique linéaire

La logique linéaire[Gir87] est une nouvelle logique constructive, issue d'une réflexion de J.-Y. Girard quant au besoin d'améliorer la représentation des démonstrations. Elle peut se comprendre comme résultant d'une analyse fine du rôle joué par les règles structurelles dans le calcul des séquents de Gentzen[Gen55]. Ces règles, considérées traditionnellement comme secondaires, spécifient que les séquences de formules apparaissant dans les séquents peuvent être traitées comme des (multi)ensembles. Elles sont au nombre de trois dans le cas de la logique intuitionniste (et, par symétrie, au nombre de six dans le cas de la logique classique[1]) :

$$\frac{\Gamma \vdash C}{\Gamma, A \vdash C}\ \text{(Affaiblissement)} \qquad \frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C}\ \text{(Contraction)}$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \text{(Echange)}$$

Ces règles sont pourvues d'un contenu logique important : la règle d'affaiblissement précise que certaines hypothèses peuvent ne pas être employées au cours d'une dérivation ; de manière semblable, la règle de contraction spécifie que toute hypothèse peut être employée un nombre illimité de fois ; quant à la règle d'échange, elle stipule qu'il n'existe aucun ordre entre les hypothèses. Aussi, l'adoption des règles structurelles au sein d'un calcul des séquents conditionne-t-elle fortement les propriétés du système logique spécifié. Par exemple, dans les formulations dues à Gentzen de la logique classique ou intuitionniste, la seule règle de contraction a pour conséquence la non-décidabilité du calcul des prédicats. Quant à l'emploi des règles d'affaiblissement et de contraction à droite dans le cas de la logique classique, il est responsable des aspects non constructifs de cette dernière.

Dans le cadre de cette analyse, la logique linéaire peut être comprise comme un système conciliant le constructivisme de la logique intuitionniste et la symétrie de la logique classique.

1. Dans le cas de la logique intuitionniste, un séquent est composé d'une suite d'hypothèses et d'une conclusion unique. En logique classique, les séquents sont symétriques. Ils sont donc formés d'une suite d'hypothèses et d'une suite de conclusions, l'interprétation de ces dernières étant disjonctive.

[Gir87] J.-Y. Girard, « Linear Logic », *Theoretical Computer Science 50*, 1987, p. 1–102.

[Gen55] G. Gentzen, *Recherches sur la déduction logique (Untersuchungen über das logische schliessen)*, Presses Universitaires de France, 1955, Traduction et commentaire par R. Feys et J. Ladrière.

	logique linéaire propositionnelle			
	logique linéaire rudimentaire			
	négation	multiplicatifs	additifs	exponentielles
négation	$A^\perp$			
conjonction		$A \otimes B$	$A \,\&\, B$	
disjonction		$A \parr B$	$A \oplus B$	
implication		$A \multimap B$		
constantes		$\mathbf{1}$, $\perp$	$\top$, $\mathbf{0}$	
modalités				$!A$, $?A$

Table 1 : *Les connecteurs de la logique linéaire*

Tout comme en logique intuitionniste, le caractère constructif est obtenu en rejetant l'usage des règles d'affaiblissement et de contraction dans la partie droite du séquent. Mais ce faisant, afin de conserver un système symétrique, on rejette également l'usage de ces mêmes règles dans la partie gauche.

Le système résultant, appelé *logique linéaire rudimentaire* (voir Table 1), présente de nombreuses propriétés intéressantes. Il est pourvu de quatre connecteurs (deux conjonctions et deux disjonctions) et de quatre constantes correspondant aux éléments neutres de ces premiers. Il est complètement symétrique, bien que constructif, et est pourvu d'une négation involutive. De ce fait, il obéit à des lois similaires à celles de De Morgan. L'implication linéaire, par exemple, n'est pas une primitive. Elle peut se définir, comme en logique classique, de la manière suivante : $A \multimap B \;=\; A^\perp \parr B$.

En logique linéaire rudimentaire, toute hypothèse doit être utilisée une et une seule fois au cours d'une dérivation. Cette propriété, qui permet de considérer la logique linéaire comme un calcul de ressources, est due, comme nous l'avons vu, au rejet des règles structurelles. Cependant, l'absence totale de celles-ci implique également que la logique linéaire rudimentaire est un système beaucoup plus faible que la logique intuitionniste ou classique. Aussi, afin de restaurer la puissance de ces dernières, est-il nécessaire d'ajouter au système des opérateurs permettant de récupérer le contenu logique des règles d'affaiblissement et de contraction. Ceci est réalisé, dans le cas de la logique linéaire complète, par le biais de deux modalités permettant un usage limité des règles structurelles, respectivement à gauche et à droite du séquent. La logique linéaire ne nie donc pas l'utilité des règles structurelles mais en souligne, au contraire, l'importance logique. De ce fait, elle les rejette en tant que règles épithéoriques[Cur77] afin de pouvoir les incorporer dans son système comme règles logiques contrôlées par l'utilisation de nouveaux connecteurs. C'est cette idée originale qui confère à la logique linéaire toute sa finesse et sa puissance.

Le projet Calligramme s'intéresse à la logique linéaire en tant qu'outil d'analyse. Cette logique permet une décomposition fine des connecteurs ; elle peut être interprétée en tant que calcul sensible à la notion de ressource ; elle présente un parallélisme inhérent ; elle rend possible

[Cur77] H. Curry, *Foundations of mathematical logic*, Dover Publications, 1977.

une étude géométrique de la notion même de preuve, par le biais de la théorie des réseaux de démonstration.

Cet intérêt porte sur quatre modules ou actions de recherche : calcul des séquents, réseaux de démonstration et d'interaction ; logique linéaire et parallélisme ; logique linéaire et grammaires formelles ; λ-calculs typés et sémantiques.

Ces quatre actions présentent à la fois un caractère théorique et appliqué. Cependant, du point de vue strictement applicatif, l'action « Logique linéaire et grammaires formelles » est la plus saillante. Un des objectifs premiers du projet Calligramme est le développement et la mise au point d'un modèle linguistique computationnel, basé sur la logique linéaire.

3.2 Calcul des séquents, réseaux de démonstration et d'interaction

Mots clés : calcul des séquents, réseau de démonstration, réseau d'interaction.

Participants : Philippe de Groote, François Lamarche, Jean-Yves Marion, Guy Perrier.

Glossaire :

Réseaux de démonstration Représentations graphiques (au sens de la théorie des graphes) des démonstrations de la logique linéaire.

Réseaux d'interaction Modèle universel de calcul basé sur une généralisation du processus d'élimination des coupures au sein des réseaux de démonstration.

Résumé :

Le but de cette action est d'étudier la théorie des réseaux de démonstration et d'interaction, et de contribuer à son développement. Nous nous intéressons à la généralisation du concept de connecteur multiplicatif, en particulier dans le cas non commutatif (réseaux ordonnés, calculs de Lambek et d'Abrusci). Une autre direction de recherche concerne l'addition des opérateurs de la logique linéaire ordinaire (quantificateurs, additifs et modalités) aux réseaux multiplicatifs. Les réseaux d'interaction, une variante des réseaux multiplicatifs, beaucoup plus lâche du point de vue du typage, retiennent également notre attention.

La théorie des réseaux est en fait un de nos outils principaux, et ses techniques sont employées dans les autres actions de recherche.

Les réseaux multiplicatifs

Le cœur de la théorie des réseaux de démonstration[Gir87] correspond au fragment multiplicatif de la logique linéaire.

Dans un premier temps, on définit la notion de structure de démonstration. Etant donné un séquent de la logique linéaire, une structure de démonstration est un graphe comprenant deux composantes :

– d'une part, la forêt des arbres syntaxiques des formules constituant le séquent ;

[Gir87] J.-Y. Girard, « Linear Logic », *Theoretical Computer Science 50*, 1987, p. 1–102.

– d'autre part, un couplage entre les feuilles de cette forêt ; ce couplage, qui correspond aux axiomes d'une démonstration séquentielle, fait correspondre à chaque occurrence positive d'une proposition atomique une occurrence négative de cette même proposition.

Dans un deuxième temps, on distingue parmi les structures de démonstration, les réseaux. Ceux-ci correspondent à des démonstrations correctes. Cette distinction s'opère au moyen de critères géométriques globaux. C'est là tout l'intérêt de cette théorie des réseaux, qui éclaire d'une lumière nouvelle la notion de démonstration.

La découverte de nouveaux critères de correction reste un thème de recherche important. Certains critères sont mieux adaptés que d'autres à telle ou telle application. En particulier, en démonstration automatique, les critères de correction peuvent être utilisés comme invariants au cours de la construction inductive d'une démonstration.

La théorie des réseaux de démonstration présente également un caractère dynamique : l'élimination des coupures. Cette dynamique correspond à une notion de normalisation (ou d'évaluation) apparentée à la notion de réduction β du lambda calcul.

Les variantes multiplicatives

La logique linéaire, du fait de sa grande souplesse, permet de nombreuses variations. Par exemple en rejetant (partiellement) la règle structurelle d'échange, on obtient des variantes (partiellement) non commutatives.

Il est également possible de se restreindre à des séquents intuitionnistes, ce qui correspond à une vision fonctionnelle (en termes d'entrées/sortie) de la logique linéaire. En particulier, le calcul syntaxique de Lambek [Lam58,Lam61] qui joue un rôle central en théorie des grammaires catégorielles, correspond exactement au fragment intuitionniste, non commutatif de la logique linéaire multiplicative. Citons également d'autres variantes qui nous intéressent au premier chef : le calcul non commutatif d'Abrusci [Abr91] et le calcul ordonné de Retoré [Ret93]. Pour chacune de ces variantes, nous étudions et développons la théorie des réseaux de démonstration associée.

Extension aux autres connecteurs

Dès que l'on s'éloigne du fragment multiplicatif, de nombreuses difficultés émergent (caractère non-local des réductions dû à la présence des boîtes, non-confluence due aux règles de commutation, etc.). De ce fait, en dehors du fragment multiplicatif, la théorie des réseaux correspond plus à un domaine de recherche actif qu'à une théorie établie.

Or le pouvoir d'expression du fragment multiplicatif est trop faible pour de nombreuses

[Lam58] J. Lambek, « The mathematics of sentence structure », *Amer. Math. Monthly 65*, 1958, p. 154–170.

[Lam61] J. Lambek, « On the calculus of syntactic types », *in : Studies of Language and its Mathematical Aspects*, Proc. of the 12th Symp. Appl. Math.., p. 166–178, Providence, 1961.

[Abr91] M. Abrusci, « Phase semantics and sequent calculus for pure non-commutative classical linear logic », *Journal of Symbolic Logic 56*, 4, 1991, p. 1403–1451.

[Ret93] C. Retoré, *Réseaux et Séquents Ordonnés*, Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, 1993.

applications. Dans le cadre de l'isomorphisme de Curry-Howard [How80], par exemple, il est indispensable d'utiliser les exponentielles si l'on veut récupérer la puissance de la logique intuitionniste et donc, du λ-calcul. Les additifs, quant à eux, permettent d'exprimer des opérateurs de choix. Finalement, les quantificateurs jouent également un rôle important dans de nombreuses applications, qu'il s'agisse des quantificateurs du premier ordre permettant d'exprimer des dépendances, ou des quantificateurs du second ordre permettant d'exprimer le polymorphisme.

Les réseaux d'interaction

Les réseaux d'interaction [Laf90] sont un modèle de calcul directement inspiré des réseaux de démonstration. Leur mécanisme de calcul est une généralisation du mécanisme d'élimination des coupures du fragment multiplicatif. Les réseaux d'interaction ne représentent pas, en général, des démonstrations. Leur théorie correspond plutôt à celle d'une machine abstraite permettant d'implanter, entre autres, les réseaux de démonstration. Du fait qu'ils correspondent à un modèle de calcul parallèle, les réseaux d'interaction jouent un rôle important dans notre étude des aspects parallèles de la logique linéaire.

3.3 Logique linéaire et parallélisme

Mots clés : calcul parallèle, calcul de processus.

Participants : Philippe de Groote, Guy Perrier.

Résumé :

Certains aspects de la logique linéaire sont intrinsèquement liés au parallélisme. Elle permet, d'une part, une meilleure compréhension des processus de parallélisation de l'exécution d'un programme séquentiel. D'autre part, elle jette un regard nouveau sur l'interprétation logique que l'on peut donner à des concepts tels que la composition parallèle, la communication entre processus, l'indéterminisme, etc.

Deux paradigmes opérationnels

La réflexion sur la pertinence de la logique intuitionniste en informatique a donné lieu à deux paradigmes permettant d'interpréter une logique constructive donnée de manière calculatoire.

Le premier de ces paradigmes, que nous appellerons le *paradigme fonctionnel*, est désigné en anglais par l'expression *proofs-as-programs*. Il est basé sur l'isomorphisme de Curry-Howard [How80] qui établit une correspondance entre λ-calcul typé et logique intuitionniste et

[How80] W. Howard, « The formulae-as-types notion of construction », *in : to H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. P. Seldin et J. R. Hindley (éditeurs), Academic Press, 1980, p. 479–490.

[Laf90] Y. Lafont, « Interaction nets », *in : Proceedings of the 17th ACM symposium on Principles of Programming Languages*, ACM Press, 1990.

ce, à trois niveaux : aux types, correspondent les formules ; aux termes, correspondent les démonstrations ; au processus de réduction des termes, correspond le processus de normalisation des démonstrations.

Le second de ces paradigmes, que nous appellerons le *paradigme logique*, correspond à l'expression anglaise *proof-search-as-computation*. Ici, ce n'est plus une démonstration qui correspond à un programme, mais un ensemble de formules ; son exécution, en présence d'une formule logique appelée requête, consiste alors à construire une démonstration normale du séquent intuitionniste dont les hypothèses correspondent au programme et la conclusion à la requête. Ce paradigme permet de définir de manière abstraite la notion de langage de programmation logique [MNPS91].

Qu'il s'agisse de l'un ou de l'autre de ces deux paradigmes, la logique linéaire présente de nombreux aspects opérationnels liés au parallélisme.

Le paradigme fonctionnel

Il est possible de coder les λ-termes (ou, plus généralement, les programmes fonctionnels) sous la forme de réseaux de démonstration [Dan90]. Cette représentation est telle que l'évaluation d'un λ-terme donné revient à éliminer les coupures au sein du réseau correspondant. Or les différentes coupures existant dans un réseau sont indépendantes les unes des autres et se réduisent de manière essentiellement locale, ce qui ouvre la porte à une parallélisation de l'exécution.

Cette observation est à la base du formalisme des réseaux d'interaction [Laf90]. Ceux-ci correspondent à un modèle universel de calcul dont la dynamique s'exprime en terme de réécritures locales et indépendantes au sein d'un graphe. Un typage très souple assure la cohérence des calculs quel que soit l'ordre dans lequel ils sont effectués.

Il est à noter que les travaux exploitant les possibilités de parallélisation de l'élimination des coupures en logique linéaire s'inscrivent pour la plupart dans un cadre intuitionniste, donc fonctionnel. Une question fondamentale reste ouverte : quel est le paradigme calculatoire sous-jacent à la logique linéaire classique ?

[MNPS91] D. Miller, G. Nadathur, F. Pfenning, A. Scedrov, « Uniform proofs as a foundation for logic programming », *Annals of Pure and Applied Logic 51*, 1991, p. 125–157.

[Dan90] V. Danos, *Une application de la logique linéaire à l'étude des processus de normalisation et principalement du lambda calcul*, Thèse de doctorat, Université de Paris VII, 1990.

[Laf90] Y. Lafont, « Interaction nets », *in : Proceedings of the 17th ACM symposium on Principles of Programming Languages*, ACM Press, 1990.

Le paradigme logique

J.-M. Andreoli et R. Pareschi [AP91], D. Miller [Mil93], N. Kobayashi et A. Yonezawa [KY92] ainsi que G. Perrier [Per95] ont montré comment on pouvait concevoir un langage de programmation parallèle reposant sur la construction de démonstrations en logique linéaire. Les formules logiques représentent des processus et la construction d'une démonstration est alors vue comme l'exécution d'un calcul de processus.

Ces travaux permettent d'interpréter les connecteurs de la logique linéaire en termes d'opérateurs de composition de processus : parallélisme, communication asynchrone, non-déterminisme interne, non-déterminisme externe, etc.

La question qui se pose alors est double. Que peut nous apprendre la logique linéaire sur les calculs de processus? Que peuvent nous apprendre les calculs de processus sur la logique linéaire ?

3.4 Logique linéaire et grammaires formelles

Mots clés : grammaire catégorielle, analyse syntaxique des langues naturelles, sémantique des langues naturelles.

Participants : Philippe de Groote, François Lamarche, Alain Lecomte, Guy Perrier.

Glossaire :
Grammaire catégorielle formalisme logique basé sur une notion fonctorielle de catégorie syntaxique, permettant la description formelle de langages.

Résumé :
Le calcul syntaxique de Lambek, qui joue un rôle central dans la théorie des grammaires catégorielles, apparaît a posteriori comme un fragment de la logique linéaire. De ce fait, celle-ci fournit un cadre mathématique permettant d'étendre ledit calcul et la notion de grammaire catégorielle. Le but de cette recherche est le développement d'un modèle de linguistique computationnelle plus souple et plus efficace que les modèles catégoriels existant actuellement.

Logique linéaire et grammaires catégorielles

La pertinence de la logique linéaire en ce qui concerne le traitement de la langue tient à sa sensibilité à la notion de ressource. Un langage (naturel ou formel) peut, en effet, être

[AP91] J.-M. Andreoli, R. Pareschi, « Linear objects: Logical processes with built-in inheritance », *New Generation Computing 9*, 3-4, 1991.

[Mil93] D. Miller, « The π-calculus as a theory in linear logic: Preliminary results », *in : Proceedings of the 1992 Workshop on Extensions to Logic Programming*, E. Lamma, P. Mello (éditeurs), *Lecture Notes in Computer Science, 660*, Springer-Verlag, p. 242–265, 1993.

[KY92] N. Kobayashi, A. Yonezawa, « Asynchronous communication model based on linear logic », *in : proceedings of Joint Intl. Conf. and Symp. on Logic Programming (JICSLP'92), Workshop on Linear Logic and Logic Programming*, Washington, November 1992.

[Per95] G. Perrier, *De la construction de preuves à la programmation parallèle en logique linéaire*, Thèse de doctorat, Université Henri Poincaré, Nancy, janvier 1995.

interprété comme un système de ressources. Par exemple, une phrase telle que :

$$*\textit{il Pierre présente à Marie à Paul}$$

est incorrecte parce qu'elle viole un principe sous-jacent aux langues naturelles selon lequel les valences verbales doivent être réalisées une et une seule fois. Les grammaires catégorielles formalisent cette idée en spécifiant qu'un verbe tel que *présente* est une ressource qui donnera une phrase p en présence d'un syntagme nominal sujet sn, d'un syntagme nominal objet sn, et d'un objet indirect $Acomp$. Ceci donne lieu à l'assignation de type qui suit :

$$\begin{array}{cccc} \textit{Pierre}: sn; & \textit{présente}: ((sn \setminus p)/Acomp)/sn; & \\ \textit{Marie}: sn; & \textit{à}: Acomp/sn; & \textit{Paul}: sn. \end{array}$$

où l'oblique ($/$) et la contre-oblique ($\setminus$) s'interprètent respectivement comme des paires de fractions se simplifiant à gauche et à droite :

$$sn \cdot ((sn \setminus p)/Acomp)/sn \cdot sn \cdot Acomp/sn \cdot sn = p$$

Cependant, on s'aperçoit très vite que ce schéma de simplification, qui est à la base des grammaires de Bar-Hillel [BH50], n'est pas suffisant. Par exemple, l'assignation

$$\textit{qui}: (n \setminus n)/(sn \setminus p),$$

qui interprète le pronom relatif *qui* comme une ressource transformant une phrase sans sujet $(sn \setminus p)$ en un modificateur à droite de nom $(n \setminus n)$, nécessite d'autres principes pour être mise en œuvre. Lambek résout ce problème en proposant d'interpréter les obliques et contre-obliques comme connecteurs implicatifs [Lam58,Lam61]. Ceux-ci obéissent alors à la loi du *modus ponens*, qui n'est autre que le schéma de simplification de Bar-Hillel :

$$\frac{\Gamma \vdash A \quad \Delta \vdash A \setminus B}{\Gamma,\Delta \vdash B} \qquad \frac{\Gamma \vdash B/A \quad \Delta \vdash A}{\Gamma,\Delta \vdash B}$$

mais également à des règles d'introduction:

$$\frac{A,\Gamma \vdash B}{\Gamma \vdash A \setminus B} \qquad \frac{\Gamma,A \vdash B}{\Gamma \vdash B/A}$$

Le calcul de Lambek a également ses limites. Il ne permet pas, entre autres, de rendre compte de phénomènes syntaxiques tels que l'extraction médiane ou les dépendances croisées. Se pose alors la question : comment étendre le calcul syntaxique de Lambek ? C'est ici qu'intervient la logique linéaire en offrant un cadre mathématique adéquat, au sein duquel il est possible de s'attaquer à cette question. En particulier, les réseaux de démonstration apparaissent comme la structure d'analyse syntaxique la plus adaptée à l'approche catégorielle.

[BH50] Y. Bar-Hillel, « A quasi-arithmetical notation for syntactic description », *Language 29*, 1950, p. 47–58.

[Lam58] J. Lambek, « The mathematics of sentence structure », *Amer. Math. Monthly 65*, 1958, p. 154–170.

[Lam61] J. Lambek, « On the calculus of syntactic types », *in : Studies of Language and its Mathematical Aspects*, Proc. of the 12th Symp. Appl. Math.., p. 166–178, Providence, 1961.

Construction d'un modèle linguistique

De la logique linéaire à un modèle linguistique, c'est-à-dire, un modèle computationnel adapté au traitement automatique de la langue, il reste une distance importante à franchir.

La construction d'un tel modèle est un des objectifs premiers du projet Calligramme. Outre les questions purement techniques liées, par exemple, au choix de tel ou tel fragment de la logique linéaire, se posent également de nombreuses questions extra-logiques. Citons, entre autres, les limites entre morphologie, syntaxe et sémantique, l'opposition entre lexique et grammaire, l'opposition entre ordre des mots et ordre d'évaluation, le choix des catégories syntaxiques de base, etc.

A terme, l'élaboration du modèle linguistique donnera lieu à l'implémentation d'un analyseur syntaxico-sémantique adapté au français. Cet analyseur est vu comme un outil générique permettant le prototypage rapide de diverses applications.

3.5 Lambda calculs typés, sémantique, complexité

Mots clés : lambda-calcul, théorie des types, sémantique dénotationnelle, sémantique des jeux, complexité implicite, complexité en information..

Participants : Guillaume Bonfante, Adam Cichon, Philippe de Groote, François Lamarche, Jean-Yves Marion, Catherine Pilière.

Résumé : *Jusqu'à l'invention des réseaux de démonstration, l'outil principal de représentation des démonstrations dans les logiques constructives était le lambda-calcul. Il n'est pas surprenant que certains aspects de la théorie du lambda-calcul et de ses extensions continuent à intéresser les membres du groupe : citons le lien existant entre la logique classique et les opérateurs de contrôle séquentiel (par le biais du lambda-mu calcul), l'étude du polymorphisme, et l'emploi de systèmes de types pour l'expression de classes de complexité. Nous utilisons également la capacité de la sémantique dénotationelle à inspirer des progrès syntaxiques. En particulier, la sémantique des jeux, qui est liée directement à la théorie des réseaux de démonstration, retient notre attention.*

Logique classique et opérateurs de contrôle

L'isomorphisme de Curry-Howard, qui doit son existence au caractère fonctionnel de la logique intuitionniste, peut être étendu à des fragments de la logique classique. En fait, certaines constructions qu'on rencontre dans les langages de programmation fonctionnels, tels les opérateurs de contrôle, n'ont pu être expliquées que par l'emploi de règles de déduction qui s'apparentent à la preuve par contradiction [Gri90].

Cette extension de l'isomorphisme de Curry-Howard à la logique classique, qui ne suit pas une voie tracée d'avance, reste présente comme thème de recherche au sein du projet.

[Gri90] T. G. Griffin, « A formulae-as-types notion of control », *in : Conference record of the seventeenth annual ACM symposium on Principles of Programming Languages*, p. 47–58, 1990.

Bien que cela reste encore spéculatif, il est à noter que l'emploi que fait Montague du λ-calcul en sémantique des langues naturels [Mon73] présente de nombreuses analogies avec la notion de continuation et de contrôle.

Polymorphisme

L'interprétation informatique des quantificateurs du second ordre donne lieu à des programmes polymorphes, c'est-à-dire, des programmes prenant certains types de données comme paramètres. Cette idée, au niveau des grammaires catégorielles, a été promue et exploitée par M. Emms [Emm89]: le polymorphisme s'avère utile pour exprimer le type d'un mot comme « et », qui peut coordonner plusieurs sortes de syntagmes.

En général, la logique intuitionniste et la logique linéaire (même restreinte au fragment correspondant au calcul de Lambek) sont indécidables au second ordre. Un thème de recherche important consiste à découvrir des fragments décidables, dont le pouvoir d'expression est suffisant pour telle ou telle application.

Sémantique dénotationnelle et sémantique des jeux

La sémantique dénotationnelle est la recherche d'interprétations de la syntaxe dans l'univers des mathématiques « ordinaires », soit les ensembles, relations et fonctions. Son importance dans le développement de la théorie des types modernes a été très grande, et ceci depuis les premiers modèles, dus à Scott, qui datent de la fin des années soixante. En particulier l'invention de la logique linéaire est due à l'étude poussée par Girard d'un modèle particulièrement simple du lambda-calcul, les espaces cohérents.

La sémantique des jeux est un nouveau type de sémantique dénotationnelle qui a la capacité de conserver beaucoup plus d'information syntaxique que les sémantiques traditionnelles. L'idée de base est très simple : on considère un type comme un jeu à deux personnes, et un terme/élément qui habite ce type comme une stratégie de jeu pour un des joueurs. Si certaines versions de la sémantique des jeux sont conçues spécifiquement pour le lambda-calcul, il reste néanmoins que le terrain naturel pour en exprimer les constructions de base est la logique linéaire, vu le rapport très étroit avec les réseaux de démonstration. La sémantique des jeux apporte certaines intuitions au niveau opérationnel, et permet par exemple la conception de machines abstraites pour le lambda-calcul. Dans une direction similaire, elle ouvre aussi des perspectives pour la modélisation de l'analyse syntaxique pour les formalismes catégoriels.

Complexité implicite des calculs

La genèse de la complexité implicite des calculs (que nous abrègerons CIC dans la suite) remonte d'une part aux années 30, lorsque la notion de calculabilité a été découverte, et d'autre

[Mon73] R. Montague, « The proper treatment of quantification in ordinary English », *in : Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics*, J. Hintikka, J. Moravcsik, P. Suppes (éditeurs), Reidel, Dordrecht, 1973.

[Emm89] M. Emms, « Parsing with Polymorphic Quantifiers », *in : Proceedings of the Seventh Amsterdam Colloquium*, M. Torenvliet (éditeur), Institute for Language, Logic and Information, Amsterdam, 1989.

part aux années 70 lorsque la théorie de la complexité a pris son essort. La CIC étudie la structure de la complexité algorithmique, qui est le plus souvent mesurée en temps ou en espace nécessaire à un calcul. Des critères syntaxiques applicables aux algorithmes d'une classe donnée (primitive récursive, λ-calcul, système T) ont été développés dans cette optique. Ils permettent d'assurer le bon comportement d'un algorithme en terme d'utilisation de ses ressources et possèdent la propriété de complétude en ce sens que, pour ne prendre que cet exemple, toutes les fonctions calculables en temps polynômial peuvent effectivement soustraire aux critères caractérisant la classe des algorithmes polynômiaux. L'engouement rencontré par cette approche s'explique en grande partie par l'apparente simplicité des critères proposés qui s'appuient essentiellement sur la *différenciation du rôle des données* dans un calcul. Cette notion a été découverte, d'une part par Harold Simmons [Sim88] et d'autre part Stephen Bellantoni & Stephen Cook [BC92], et Daniel Leivant [Lei94]. La séduction de cette théorie se reflète, aussi, dans ses origines car ces découvertes suggèrent un lien délicat entre la calculabilité classique, qui a des fondements mathématiques stables, et la notion de calcul.

De nombreuses caractérisations de classes de complexité ont été récemment obtenues [LM93a, LM93b,LM94] [11]. La fécondité de cette approche s'explique par l'aspect purement syntaxique, et donc décidable, des conditions imposées à un algorithme pour garantir son appartenance à une classe de complexité donnée.

Dans le cadre de la construction de logiciel certifié, généralement obtenu par l'isomorphisme "proofs as programms", la complexité du programme extrait d'une preuve doit être bornée. A quoi servirait un programme sûr qui ne terminerait jamais par manque ou par épuisement de ses ressources de calcul? Pour cette raison, une tentative de réponse est apportée dans le système NUPRL. L'apport de la CIC est un cadre pour *raisonner sur la complexité des calculs.*

Théorie algorithmique de l'information

La *théorie algorithmique de l'information* (AIT), connue aussi sous le nom de complexité de Kolmogorov, est actuellement l'objet d'un grand engouement de chercheurs venant de nombreux pays et de nombreuses disciplines. Nous avons organisé les troisièmes journées sur l'AIT au LORIA en 1999. La complexité de Kolmogorov d'un objet (d'une séquence binaire qui le code) est la longueur d'un plus petit programme qui l'engendre ; cet entier prend dans cette

[Sim88] H. Simmons, « The Realm of Primitive Recursion », *Archive for Mathematical Logic 27*, 1988, p. 177–.

[BC92] S. Bellantoni, S. Cook, « A new recursion-theoretic characterization of the poly-time functions », *Computational Complexity 2*, 1992, p. 97–110.

[Lei94] D. Leivant, « Predicative recurrence and computational complexity I: Word recurrence and poly-time », *in : Feasible Mathematrics II*, P. Clote et J. Remmel (éditeurs), Birkhäuser, 1994.

[LM93a] D. Leivant, J.-Y. Marion, « Lambda Calculus Characterizations of Poly-Time », *in : Typed Lambda Calculi and Applications*, M. Bezem, J. Groote (éditeurs), LNCS 664, Springer-Verlag, p. 15, 1993.

[LM93b] D. Leivant, J.-Y. Marion, « Lambda Calculus Characterizations of Poly-Time », *Fundamenta Informaticae 19*, 1,2, September 1993, p. 167,184.

[LM94] D. Leivant, J.-Y. Marion, « Ramified Recurrence and Computational Complexity II: Substitution and Poly-Space », *in : Proceedings 8th Workshop on Computer Science Logic, Kazimierz (Poland)*, L. Pacholski, J. Tiuryn (éditeurs), *Lecture Notes in Computer Science, 933*, Springer Verlag, p. 486–500, septembre 1994.

théorie le sens du contenu en information de la séquence. On appelle ce type d'information *information algorithmique*. La théorie développée à partir de cette idée éclaire et enrichit de nombreux domaines (l'analyse d'algorithmes évidemment où elle est, par la méthode d'incompressibilité, un substitut aux approches combinatoires et de dénombrement, la théorie des langages, mais aussi l'apprentissage, la thermodynamique, la biologie, voire par effet de mode la gestion d'entreprises).

4 Domaines d'applications

Résumé : *Le projet Calligramme vise à appliquer ses compétences en logique linéaire au traitement automatique des langues, et plus précisément dans l'analyse syntaxique de textes (et la génération de représentations sémantiques à partir de ceux-ci), la génération automatique de textes à partir de données formelles, et la fouille de données sur des corpus complexes. Le français est la langue privilégiée.*

4.1 Les grammaires catégorielles dans la linguistique informatique

Comparées aux autres types de grammaires, les grammaires catégorielles et plus particulièrement leur utilisation grammaticale de la théorie de la démonstration, représentent un champ d'investigation neuf, et donc en pleine expansion, dans le domaine du traitement automatique des langues naturelles. Actuellement des linguistes comme Bob Carpenter, Mark Johnson, Aravind Joshi, et Carl Pollard, spécialistes de grammaires bien implantées telles HPSG, LFG, ou TAG, en viennent aux solutions qui se dégagent de l'étude logique des grammaires catégorielles, puisque les phénomènes visés échappent aux autres types de grammaires.

Une raison pratique pour l'emploi de ces formalismes logiques est qu'ils sont basés sur des propriétés universelles communes à d'autres systèmes de communication (systèmes de déduction logique, calcul de processus). Il est possible de dériver conjointement des propriétés syntaxiques, sémantiques, voire même pragmatiques avec les mêmes outils. Or une réalisation complète en traitement automatique des langues naturelles, telle la traduction automatique ou la génération de texte, se doit de manipuler simultanément ces diverses structures.

4.2 Analyse syntaxique

L'intérêt de l'analyse syntaxique est central dans le traitement automatique des langues naturelles : puisqu'elle donne accès à une représentation sémantique, elle est un ingrédient essentiel de la traduction automatique, de la génération automatique, et de l'analyse de corpus. Un analyseur syntaxique est une composante essentielle de nombreux logiciels, comme par exemple des programmes de fouilles de données, ou de réponse à des requêtes verbales sur des bases de données. La majorité des analyseurs en opération actuellement sur de tels logiciels se limite à des types de phrases simples, catalogués d'avance. Il existe un besoin pour des analyseurs plus performants au niveau de la complexité, capables de traiter par exemple des enchâssements de relatives. De tels programmes pourraient traiter directement des ouvrages techniques comme des traités scientifiques.

On fait de plus le pari que de tels analyseurs auraient à gagner à se baser sur des fondements

solides en théorie linguistique. En effet, seule une théorie linguistique solide peut envisager les faits syntaxiques dans leur globalité, c'est-à-dire non réduits à la manière dont ils occurrent dans telle ou telle langue particulière. On peut de cette manière espérer mieux traiter la question du multilinguisme. C'est la raison pour laquelle, à côté des grammaires catégorielles, la conception chomskyenne de Principes et Paramètres a retenu notre attention.

4.3 Génération automatique de textes

Si la génération de textes dans toute sa généralité n'est pas un thème de notre équipe, elle retient cependant notre attention dans certains cas précis.

Il existe plusieurs situations où l'on doit transformer des données informatiques en un langage plus accessible aux humains, par exemple si on veut produire des bulletins météorologiques. Une caractéristique commune à ces problèmes est que le vocabulaire est très spécialisé et donc limité, mais qu'il n'y a aucune limite sur le nombre et la structure des phrases qu'on peut être amené à produire.

La thèse de S. Pogodalla se situe dans ce domaine. Cette collaboration avec Xerox vise à réaliser des générateurs multilingues pour usage grand public: à partir de la sélection de quelques structures conceptuelles au moyen d'un langage graphique, générer du texte dans une langue donnée.

5 Résultats nouveaux

5.1 Calcul des séquents, réseaux de démonstration et d'interaction

Philippe de Groote a développé un nouveau critre de correction pour les structures de démonstration intuitionnistes multiplicatives [10]. Ce critère est basé sur une décoration de la structure de démonstration par les éléments d'un monoïde commutatif libre. Un des avantages de cette approche est qu'elle s'adapte facilement au cas (partiellement) non commutatif et au cas non associatif : il suffit de remplacer le monoïde commutatif par une structure librement engendrée, modulo les lois structurelles auxquelles obéit le calcul considéré.

Philippe de Groote a également développé un critère de correction pour les structures de démonstration non commutatives. Ce critère lui permet de réduire la prouvabilité de toute formule du calcul de Lambek à un problème d'analyse dans une grammaire algébrique [17]. Il est alors possible, grace à cette réduction, de dériver de nouveaux algorithmes d'analyse catégorielle.

Grace à une nouvelle forme de calcul des séquents, Philippe de Groote a montrer que le calcul non associatif de Lambek avec produit était décidable en temps polynomial [19].

François Lamarche a développé une notion originale de réseau de démonstration non associatif. Ceci lui a permis de définir une version classique du calcul non-associatif de Lambek (NL).

5.2 Logique linéaire et grammaires formelles

Les recherches de l'équipe concernant les rapports entre la logique linéaire et les grammaires formelles ont pour objectif la construction d'un nouveau modèle linguistique catégoriel.

Au cours d'une première phase expérimentale, plusieurs membres du projet ont proposé des ébauches de modèles linguistiques basés sur la logique linéaire.

Alain Lecomte et Christian Retoré ont poursuivi le développement du modèle qu'ils avaient proposé en 1995 [LR95]. Ce modèle, enrichi par des étiquettes calculables décrivant l'ordre des constituants, est maintenant développé dans le fragment intuitionniste du calcul ordonné : les réseaux considérés ont une forme bien spécifique, comportant au plus une conclusion. La communication [LR98] décrit plus formellement le type des grammaires considérées et fait le lien avec les *Tree Adjoining Grammars*.

Alain Lecomte a continué son étude des relations entre le modèle chomskyen d'analyse du langage et les logiques sensibles aux ressources. Il a en particulier montré comment donner une version "representationnaliste" du minimalisme au moyen de réseaux de preuves de la logique linéaire enrichie du connecteur précède, lors du «Fifth Roma workshop on dynamic perspectives in logic and linguistics». Dans cette version, la construction des réseaux représente la partie grammaticale proprement dite. Les interprétations phonologique et sémantique sont obtenues au moyen de la définition de chemins au sens des chemins de Lamarche qui donnent un critère de correction pour les réseaux intuitionnistes.
Cette approche représentationnaliste a été continuée en 1999. A. Lecomte a fait le lien entre les logiques hybrides (telles qu'utilisées par P. Blackburn, N. Kurtonina, M. de Rijke...) et l'approche réseaux de preuve dans la construction de preuves qui correspondent à des phrases [22]. Les logiques mélangées dans cette approche sont d'une part le calcul de Lambek classique (non commutatif) et d'autre part le calcul linéaire multiplicatif classique (commutatif). Le deuxième calcul sert à agencer des séquents du premier. On obtient finalement un système immergé dans la logique partiellement commutative de de Groote. Son avantage est qu'on peut limiter l'ensemble des preuves admises.

Alain Lecomte a d'autre part développé une version "dérivationnaliste" du même programme minimaliste (présentée à LACL'98) dans le cadre des grammaires catégorielles multimodales [Lec98]. Ce travail a abouti à des conclusions voisines de celles auxquelles sont arrivés d'autres auteurs, comme T. Cornell [Cor97] et surtout W. Vermaat, une étudiante de M. Moortgat. A. Lecomte a fait une comparaison de ces approches, présentée dans l'atelier sur les Logiques de Ressources et les Grammaires Minimalistes lors de la dernière école d'été ESSLLI (Utrecht, 1999)[21].
En 1999, il est revenu sur l'utilisation de la logique linéaire intuitionniste pour la formalisation de grammaires basées sur les Principes et Paramètres et sur le programme Minimaliste de N. Chomsky [Cho96].

[LR95] A. Lecomte, C. Retoré, « Pomset logic as an alternative categorial grammar », *in : Proceedings of the Conference of the European Summer School in Logic, Language and Information*, Barcelona, 1995.

[LR98] A. Lecomte, C. Retoré, « Words as modules: a lexicalised grammar in the framework of linear logic proof nets. », *in : Mathematical and Computational Analysis of Natural Language (Proceedings of International conference on mathematical linguistics II), Tarragone*, C. Martin-Vide (éditeur), *45*, John Benjamins, p. 129–144, juin 1998.

[Lec98] A. Lecomte, « Multimodal Logic for Syntax », *Logica Trianguli 2*, 1998.

[Cor97] T. Cornell, « A type-logical perspective on minimalist derivations », *in : Formal Grammar'97*, G. van Kruijf, R. Oehrle (éditeurs), ESSLLI'97, Aix-en-Provence, 1997.

[Cho96] N. Chomsky, *The Minimalist Program*, The MIT Press, 1996.

Avec C. Retoré, il a proposé une reconstruction logique des grammaires minimalistes (telles que formalisées par E. Stabler [Sta97], [Sta99]). L'idée de base consiste à reprendre le calcul de Lambek, mais à l'utiliser différemment. En l'occurrence, les items lexicaux sont associés à des séquents dont la partie gauche est vide. On tente de construire une démonstration de la proposition `cp` à partir de ces séquents. Les types utilisés sont construits au moyen de / et \ mais aussi du produit •. L'insertion d'un item (item lexical ou syntagme déjà construit) qui a pour type un type produit se fait en deux temps. D'abord, il faut introduire des hypothèses dans la démonstration centrale, qui correspondent chacune à un des conjoints du produit. Ensuite, on utilise (préférablement dans un système de déduction naturelle) la règle d'élimination du produit, sur la partie de preuve déjà construite et le séquent axiome correspondant au type produit. En utilisant un système étiqueté par des chaines, cela a pour conséquences d'insérer une même chaine à plusieurs points d'insertion. Cette multi-occurrence d'une même sous-chaîne correspond à l'idée de déplacement (transformation *Move*) dans le modèle chomskyen.
Ils ont ensuite montré que le calcul de Lambek ne suffisait pas car il restreint trop le type de mouvements possibles: il formalise bien les mouvements de tête en ce qu'ils ne doivent jamais se croiser, mais il est inadequat pour formaliser les cas où un mouvement de syntagme croise un mouvement de tête. Dans ces cas là, on doit ou bien introduire et gérer une modalité de permutation, analogue au connecteur d'échange d'Abrusci, ou bien se placer dans le calcul partiellement commutatif de de Groote. C'est plutôt cette deuxième alternative qui est retenue. Afin de ne pas avoir "trop" de preuves, on doit toutefois utiliser un principe d'économie, qui est la traduction en termes logiques de la condition chomskyenne du "mouvement le plus court possible".
Lecomte et Retoré [20] ont montré qu'ils obtenaient ainsi un modèle totalement équivalent aux grammaires minimalistes de Stabler, qui a donc la capacité générative des formalismes "mildly context-sensitive".

Guy Perrier a continué à développer le formalisme grammatical qu'il a proposé à partir des réseaux de démonstration de la logique linéaire intuitionniste. L'idée de départ est d'utiliser la sensibilité aux ressources de la logique linéaire pour représenter de façon uniforme sous forme d'un réseau de démonstration les dépendances entre constituants syntaxiques d'une phrase, qu'elles soient lointaines ou locales. Ce réseau sert alors de vecteur à l'information linguistique qui circule sous forme d'étiquettes pour composer la forme phonétique et la représentation sémantique de la phrase [13].
La spécificité de la logique linéaire intuitionniste permet de donner une représentation plus compacte des réseaux de démonstration à l'aide d'arbres polarisés et sous-spécifiés. La polarisation consiste à ajouter une polarité -1, 0 ou +1 à chaque noeud et la sous-spécification à indiquer qu'un noeud en domine un autre en ne précisant pas le chemin qui permettra de réaliser cette domination.
En appliquant ce résultat à la linguistique, Guy Perrier a pu faire évoluer son formalisme vers un modèle fondé sur les arbres polarisés et sous-spécifiés comme structures syntaxiques de base. Dans ce formalisme qu'il a baptisé *Grammaires d'Interaction*, la compostion syntaxique

[Sta97] E. Stabler, « Derivational minimalism », *in: Logical Aspects of Computational Linguistics*, C. Retoré (éditeur), *LNCS/LNAI, 1328*, Springer, p. 68–95, 1997.

[Sta99] E. Stabler, « Remnant movement and complexity », *rapport de recherche*, UCLA, Los Angeles, 1999.

est réalisée par branchement de noeuds duaux [28].
Il est à noter que, de façon totalement indépendante, une équipe du laboratoire de linguistique informatique de l'université de la Sarre autour de Claire Gardent travaille sur un formalisme très proche. A l'origine de ce travail, il y a un intérêt pour les descriptions d'arbres utilisées par Vijay-Shanker pour résoudre certains problèmes liés au formalisme des TAG [VS92] et une idée de Reinhard Muskens qui a consisté à ajouter des polarités aux noeuds de ces descriptions [MK98]. Une collaboration active et fructueuse a pu être entamée avec cette équipe dans le cadre d'une action intégrée Procope. Du côté français, c'est Guy Perrier qui en est responsable et du côté allemand, c'est Claire Gardent. Cette action a permis non seulement de confronter les idées théoriques mais aussi d'entamer une implémentation des Grammaires d'Interaction à l'aide du langage de programmation par contraintes Oz [Smo95].
Le travail actuel de Guy Perrier vise à raffiner le formalisme des Grammaires d'Interaction en descendant les polarités du niveau des constituants (les noeuds des arbres syntaxiques) au niveau des traits qui sont attachés à ces constituants (sous forme de matrices). La composition syntaxique est alors réalisée sous forme de superposition partielle d'arbres contrôlée par l'interaction "électrostatique" entre les traits. En plus, un système de règles est destiné à modéliser le fait que l'apparition, la rencontre de certains traits peut en engendrer de nouveaux. Le développement, en cours, d'un prototype d'analyseur syntaxique sous Oz vise à tester la pertinence linguistique et computationnelle du formalisme.

En se basant sur [dGR96], Sylvain Pogodalla a proposé une utilisation des réseaux sémantiques pour la génération de phrases dans le cadre du calcul de Lambek. Ceci permet d'établir le processus de génération comme un problème de recherche de preuve en logique linéaire multiplicative. Cette recherche de preuve s'effectue sur un réseau (dont on cherche les axiomes) contenant des coupures dont l'élimination donne un réseau connu par ailleurs. La résolution peut alors s'exprimer à l'aide d'équations matricielles et offre une caractérisation des formes sémantiques permettant une génération polynomiale.

Depuis quelques annees François Lamarche s'interroge sur le problème: « quelle est la façon correcte de de représenter une analyse linguistique du point de vue formel? ». On sait que des structures plus ou moins arborescentes sont toujours impliquées. On sait aussi qu'il existe different niveaux de représentation, disposés de façon plus ou moins hiérarchique, cette hierarchie étant souvent décrite au moyen de l'axe "structures proches de la surface / structures plus profondes". Si on peut isoler un niveau donné, la "tranche" à ce niveau, a une forme d'arbre. Mais la structure de cet arbre varie d'un niveau à l'autre, et il est certain que la robustesse de la faculté d'analyse du langage des humains (sa tolérance au bruit) provient en grande partie du fait que tous les niveaux de représentation sont traités simultanément, ce qui permet l'emploi des multiples redondances du langage pour la correction.

[VS92] K. Vijay-Shankar, « Using Description of Trees in a Tree Adjoining Grammar », *Computational Linguistics 18*, 4, 1992, p. 481–517.

[MK98] R. Muskens, E. Krahmer, « Talking about Trees and Truth-conditions », *in: Logical Aspects of Computational Linguistics, LACL'98*, Grenoble, France, Décembre 1998.

[Smo95] G. Smolka, « The Oz Programming Model », *in: Computer Science Today*, J. van Leeuwen (éditeur), *Lecture Notes in Computer Science, vol. 1000*, Springer-Verlag, Berlin, 1995, p. 324–343.

[dGR96] P. de Groote, C. Retoré, « Semantic readings of proof nets », *in: Formal Grammar*, G.-J. Kruijff, G. Morrill, D. Oehrle (éditeurs), FOLLI, p. 57–70, August 96.

Les premières conclusions qu'il a obtenues impliquaient l'emploi de l'algèbre universelle linéaire, autrement dit qui n'implique que des théories telles que deux termes dans une équation contiennent toujours exactement le meme ensemble de variables, chacune apparaissant une et une seule fois de chaque côté du terme. Dans une telle théorie la valeur d'un terme a le même statut que les variables qu'il contient, et on peut effacer la différence « entrée-sortie », ce qui peut être extrêmement profitable comme nous l'a enseigné la logique linéaire, et a du sens du point de vue linguistique. Cet état d'avancement des travaux a été présenté en août 99 a Dagstuhl.

Depuis, ses idées ont continué d'évoluer, et elles prennent maintenant la forme d'une « logique des partitions », dans laquelle il n'y a pas de différence entre un symbole fonctionnel et un symbole de prédicat. Une telle logique est une théorie du premier ordre, qui doit être vue comme la spécification d'un type de données, et les modéle finis sont les élements du type de donné. Chaque modèle de la théorie doit être vue comme un espace (ici l'exemple prototype est la théorie des algèbres de Boole, etant donné que tout modèle de cette théorie correspond a un espace topologique, selon le théorème de Stone). Une formule atomique $\phi(x_1,\ldots,x_n)$ de la théorie correspond à une partition de cet espace en n composantes, et à un symbole fonctionnel $n-1$-aire de l'algèbre universelle. Ceci est une application du principe que Girard met au coeur de la logique linéaire, c'est-à-dire qu'un objet syntaxique doit etre vu comme un espace.

Cette approche semble avoir des points de contact avec plusieurs branches de l'informatique. On peut dire entre autres qu'il s'agit d'une théorie générale des contraintes et de l'unification, au sens de la réécriture. Les rsultats préliminaires de ces travaux ont été présentés à Sarrebruck en novembre, dans le cadre de l'action intégée Gemola, et a Montréal en décembre, lors d'une rencontre de la Société Canadienne de Mathématiques.

Depuis septembre 1997, un groupe de travail dédié exclusivement à la construction du modèle linguistique se réunit chaque semaine. Son but est de faire converger les différentes propositions développées jusqu'à présent.

5.3 Lambda calculs typés, sémantique, complexité

Catherine Pilière, dans le cadre de son travail de thèse, étudie les rapports entre logique classique et opérateurs de traitement d'exceptions à la ML. Elle développe en particulier un λ-calcul simplement typé de traitement des exceptions introduit par Philippe de Groote [dG95], via l'adjonction d'un opérateur de récursion général. Après avoir établi la confluence du calcul avec point fixe [23], elle a défini pour celui-ci une sémantique dénotationnelle basée sur la translation CPS proposée dans [dG95] et possédant la propriété de *computational adequacy*. Ce dernier résultat a fait l'objet d'une présentation lors du workshop du GDR «Logique classique et programmation» qui s'est tenu à Chambéry les 20 et 21 mai 1999. Il est actuellement en cours de rédaction.

Philippe de Groote a développé une démonstration modulaire de la normalisation forte des déductions naturelles intuitionistes en présence des conversions permutatives [18]. Cette démonstration est basée sur les notions de traduction négatives et de simulation CPS.

[dG95] P. de Groote, « A simple Calculus of Exception Handling », *in : Second International Conference on Typed Lambda Calculi and Applications, TLCA'95*, M. Dezani, G. Plotkin (éditeurs), *Lecture Notes in Computer Science, 902*, Springer Verlag, p. 201–215, 1995.

G. Bonfante, A. Cichon et F. Lamarche étudient les propriétés des ordres fondés sur la notion de terme ordinal. Nous nous appuyons en premier lieu sur les travaux de Grzegorczyk [Grz53] qui établit une stratification des fonctions primitives récursives. Cette hiérarchie a été étendue par la suite par Löb et Wainer [LW71] en utilisant la notion d'*ordinal*. Le concept d'ordinal, généralisation du concept d'entier naturel, est utilisé comme outil de mesure de la "complexité" d'un ordre bien fondé.

L'idée de Löb et Wainer repose sur l'opération de diagonalisation d'une famille de fonction $(f_n)_{n\in\mathbb{N}}$. La fonction diagonalisée $(\Delta f)(n) = f_n(n)$ a une complexité plus grande que chacune des fonctions de la famille. L'intérêt est que cette construction est similaire à la construction de séquences fondamentales (suites croissantes d'ordinaux) pour les ordinaux. En ce sens, ce ne sont pas les ordinaux *per se* qui sont le bon objet d'étude mais les ordinaux équipés de séquences fondamentales. Les travaux d'Adam Cichon et Hélène Touzet [CT96] dans ce domaine ont conduit à la proposition d'un système de termes ordinaux. Vis-à-vis des ordinaux, de tels systèmes offrent davantage de perspectives de par leur plus grande souplesse d'utilisation et de par des distinctions plus fines qu'elles permettent de faire entre deux processus.

Nous poursuivons à présent ces travaux en proposant un pendant sémantique à l'approche de Cichon et Touzet qui est essentiellement syntaxique. En effet, les notations ordinales peuvent être vues comme des *ordinaux généralisés*, et l'on peut travailler avec, sans référence à la syntaxe. En outre, le procédé classique vise à construire un ensemble de manière inductive puis de plaquer un ordre sur cet ensemble. Nous penchons pour une nouvelle approche qui consiste à intégrer l'ordre en tant que tel dans le calcul. Nous pouvons donner une justification sémantique de la syntaxe proposée par Cichon et Touzet, mais également un procédé pour simplifier cette syntaxe (cf [BCL98]).

D'autre part, nous nous sommes penchés sur les travaux de Simmons [Sim93] qui propose une formalisation des termes ordinaux dans un lambda-calcul typé. Pour aller plus loin dans la construction d'ordres, nous avons cherché à utiliser une théorie des types dépendants à la Martin-Löf, théorie qui permet de manipuler à la fois la relation d'ordre et ses éléments.

Si l'idée d'utiliser la catégorie des ensemble ordonnés comme sémantique du calcul peut paraître tentante a priori, elle est décevante dans la pratique : à cause de la transitivité, les schémas d'induction sont rapidement très compliqués voire inabordables. Il est plus simple de construire des relations binaires qui sont des ordres *a posteriori*. Nous nous basons pour cela, sur une notion d'algèbre universelle fondée sur une nouvelle catégorie monoïdale symétrique fermée sur les multi-graphes. Ces travaux donnent lieu à un procédé de construction d'ordres inductifs, suffisamment général pour définir ω, les termes ordinaux ou pour donner une formalisation du théorème de Kruskal. Ces travaux sont présentés dans [24].

[Grz53] Grzegorczyk, « Some classes of recursive functions. », *Rozprawy Mate 4*, 1953, Warsaw.

[LW71] M. Löb, S. Wainer, « Hierarchies of number-theoretic functions. », *Arch. Math. Logik 14*, 1971, p. 198.

[CT96] A. Cichon, H. Touzet, « An Ordinal Calculus for Proving Termination in Term Rewriting », *in : Int. CAAP-Coll.on Trees in Algebra and Programming*, H. Kirchner (éditeur), *Lecture Notes in Computer Science, 1059*, Springer-Verlag, p. 226–240, 1996.

[BCL98] G. Bonfante, A. Cichon, F. Lamarche, « A semantics and a syntax for ordinal notations and hierarchies », *Rapport de recherche*, LORIA, 1998, version intermédiaire.

[Sim93] H. Simmons, « Logic and Computation », Personal communication, 1993.

Nous avons abouti à la définition d'un λ-calcul qui permet de manipuler ces ordres inductifs. Ce travail, en cours de rédaction, fera partie de la thèse de G. Bonfante.

Complexité implicite des calculs Dans [11], D. Leivant et JY Marion ont caractérisé la classe de fonctions ALogTime=NC^1. Une fonction est dans NC^1 si d'une part, un bit de sa sortie est obtenu par un calcul parallèle (circuits booléens uniformes) qui utilise un nombre polynômial de processeurs et dont le temps de calcul est logarithmique, et si, d'autre part, la taille de sa sortie est bornée polynômialement. Les mesures sont prises relativement à la taille de l'entrée. Pour ce faire, nous avons considéré des algorithmes construits sur un schéma de récurrence, sur les arbres, avec substitution de paramètre.

Adam Cichon a soumis un article, en collaboration avec Elias Tahhan-Bittar de l'université Simon Bolivar au Venezuela, au 11ème SLALM (XI Simposio Latinoamericano de Logica Matematica), sur une caractérisation de la hiérarchie de Grzegorczyk [16].

L'identification de la complexité algorithmique du calcul par sa preuve de terminaison est un sujet qui nous intéresse particulièrement. En effet, dans [15, 9] et [Mar98], nous avons établi une relation entre la complexité d'un système de réécriture et sa preuve de terminaison lorsque celle-ci s'appuie sur les interprétations polynômiales. Guillaume Bonfante, dans [24], s'est intéressé à l'ordre `kbo` En donnant des contraintes syntaxiques sur l'ordre, il a obtenu trois caractérisations, LINSPACE, ESPACE et ETIME. Les caractérisations en espace ont fait l'objet d'une communication au quatrième Workshop International sur la Terminaison à Dagstuhl.

Récemment, Jean-Yves Marion a proposé d'utiliser les outils de la CIC, voir 3.5 pour étudier la "qualité" d'une spécification. L'objectif est d'analyser une spécification afin de déterminer la complexité de la fonction et d'extraire un algorithme efficace. Pour être concret, illustrons le problème par un exemple. Etant donné deux chaînes de caractères $u = u_1 \cdots u_m$ et $v = v_1 \cdots v_n$ dans $\{0,1\}^*$, une sous-suite commune de longueur k est définie par deux suites d'indices $i_1 < \cdots < i_k$ et $j_1 < \cdots < j_k$ telles que $u_{i_q} = v_{j_q}$. Considérons le problème bien connu du calcul de la plus longue sous-suite entre deux mots (c'est une simplification de la commande `diff` sous Unix). Ce problème est classiquement résolu par la spécification récursive suivante :

$$\begin{array}{ll} \texttt{lcs}(\epsilon,\epsilon;) & \to 0 \\ \texttt{lcs}(i(x),i(y)) & \to s(\texttt{lcs}(x,y)) \\ \texttt{lcs}(i(x),j(y)) & \to \texttt{max}(\texttt{lcs}(x,j(y)),\texttt{lcs}(i(x),y)) \qquad i \neq j \end{array}$$

Il n'est pas trop difficile de se convaincre que, en tant qu'algorithme récursif cette définition résout le problème en temps exponentiel. Or, il est bien connu qu'il existe un algorithme de type programmation dynamique qui résout ce problème efficacement en temps polynomial. Est-il possible de synthétiser semi-automatiquement le bon algorithme à partir de cette spécification? Une restriction du très classique ordre de terminaison *Multiset path ordering*, proposé dans [27] et qui repose sur la théorie de l'analyse prédicative de la récursion, permet d'établir que la fonction décrite par `lcs` est calculable en temps polynomial. De plus et surtout, un

[Mar98] J.-Y. Marion, « An hierarchy of terminating algorithms with semantic », *Rapport de recherche*, LORIA, 1998.

bon algorithme est automatiquement construit. De même, avec Adam Cichon, nous avons proposé [25] une restriction similaire de l'ordre *Lexicographic Path Ordering* qui analyse d'autres formes d'algorithmes.

A moyen terme, l'objectif est de construire des prototypes d'analyseur/compilateur à partir d'analyse statique de programme qui s'appuie sur des versions prédicatives des ordres de terminaison ou de systèmes de types, comme sus-mentionnées. Ces prototypes devront analyser la complexité d'une fonction à partir d'une spécification puis synthétiser un algorithme efficace, le cas échéant. Jean-Yves Marion organisera un workshop satellite à LICS en Juin 2000 sur ce thème.

Théorie algorithmique de l'information

Serge Grigorieff et Jean-Yves Marion [26] ont étudié la notion d'information d'un objet produit par un calcul non-déterministe. Nous avons proposé des conditions suffisantes pour définir une notion d'entropie non-déterministe. Jérôme Besombes a commencé son travail de thèse en Septembre, il étudie actuellement l'algorithme optimal pour inverser les fonctions, proposé par L. Levin. Il devrait poursuivre sa thèse sur l'apprentissage des grammaires catégorielles en utilisant la complexité en information.

6 Contrats industriels (nationaux, européens et internationaux)

6.1 Collaboration avec Xerox

Le projet Calligramme entretient, depuis sa création, des relations avec le centre de recherche de Xerox à Grenoble. Deux membres de la société Xerox, Marc Dymetman et Aarne Ranta, ont ainsi participé aux Journées Linguistiques organisées à Nancy du 18 au 20 mars 1998. Ces relations ont également donné lieu à un soutien financier de la conférence LACL'98, de la part de Xerox.

Leur collaboration s'est concrétisée en 1998 par l'inscription en thèse sur contrat CIFRE de Sylvain Pogodalla.

Sylvain Pogodalla est maintenant bien intégré comme ingénieur à XRCE. Il a donné de nombreux exposés sur l'état d'avancement de sa thèse. Il jouit d'un financement spécifique lui permettant des déplacements à Nancy et à Rennes (où il peut travailler avec son co-directeur de thèse Christian Retoré).

7 Actions régionales, nationales et internationales

7.1 Actions nationales

Jean-Yves Marion est membre du PRC AMI (CNRS), Complexité, Modèles finis et Arithmétiques faibles.

Jean-Yves Marion est membre du PRC AMI (CNRS), Complexité de Kolmogorov.

7.2 Actions européennes

Calligramme a été à l'initiative de la soumission d'un projet pour le programme européen IST qui n'a pas abouti. Ce projet visait à développer un environnement de description des langues pour la traduction automatique et la préparation de documents multilingues. Il regroupait trois PME européennes, Neurosoft en France, Compris.com en Allemagne et Incyta en Espagne, ainsi que trois partenaires académiques : une équipe de l'université de Catalogne à Barcelone autour de Glynn Morrill, une de l'université de Poznan en Pologne autour de Wojciech Buszkowski et Calligramme qui devait être le coordinateur du projet. Les échanges qui ont permis de mettre sur pied ce projet ont été très positifs et sont un premier pas vers une collaboration de Calligramme avec des PME dans le domaine du traitement des langues.

Calligramme participe à un réseau TMR sur le thème de la logique linéaire. Ce réseau regroupe des équipes de Marseille (Institut de mathématiques de Luminy et projet Calligramme rattaché à Marseille), Bologne (Universités de Bologne, Turin et Udine), Cambridge (Universités de Cambridge et Oxford), Edinbourg (Universités d'Edinbourg et d'Aarhus), Lisbonne (Université de Lisbonne et Université Nouvelle de Lisbonne), Paris (LIPN et DMI, ENS) et Rome (Universités de Rome, Pise, Bari, Sienne, et CNR-IAC).

7.3 Actions internationales

Les relations scientifiques qu'entretient Calligramme avec diverses universités ou instituts étrangers ont abouti à plusieurs actions intégrées bilatérales:

- Projet Galileo avec l'Université de Rome 3 (Michele Abrusci),
- Projet Van Gogh avec l'Université d'Utrecht (Michael Moortgat),
- Projet Picasso avec l'Université Polytechnique de Catalogne (Glyn Morrill),
- Projet Procope avec l'Université de la Sarre (Claire Gardent),
- Projet Alliance avec l'Université de Birmingham (Valeria de Paiva).

7.4 Visites et invitations de chercheurs

Adam Cichon a effectué un séjour de trois semaines, du 28 août 1999 au 19 septembre 1999, à l'Université Simon Bolivar au Venezuela comme professeur invité. Il a donné un cours de 12h sur la théorie de la démonstration et deux séminaires. Le but de la visite était de maintenir la collaboration avec E. Tahhan.

Elias Tahhan a effectué un séjour d'un mois (juin-juillet 1999) à Nancy en tant que professeur invité INRIA. Le but de la visite etait de collaborer avec A. Cichon.

Dans le cadre de l'action Procope "GeMoLa", Guy Perrier a reçu Denys Duchier, Claire Gardent, Alexander Koller et Stephan Thater de l'université de Sarrebrück les 15 et 16 mars 1999.

8 Diffusion de résultats

8.1 Animation de la Communauté scientifique

Administrative

François Lamarche est President du comite des bourses Inria.

Worskshop et groupe de travail

Adam Cichon est membre du comité de programme de «International Workshop on Termination».

François Lamarche a organisé un workshop Van Gogh de trois jours en Avril, une quinzaine de personnes sont venues d'Angleterre, de Hollande, et d'Espagne.

Jean-Yves Marion a organisé les troisièmes journées sur la théorie algorithmique de l'information, TAI'99, 20 et 21 Mai 1999, LORIA, Nancy. Ces journées ont réuni des orateurs prestigieux, citons entre autre L. Levin, A. Verashagin, A. Shen, P. Vitanyi. Une quarantaine de participants sont venues d'Allemagne, Etats-unis, Hollande, Japon, Russie. Les journées ont été subventionées par : Loria, CNRS, GDR-ALP, Université Nancy 1, Nancy 2, INPL, le grand Nancy et la région Lorraine. Le comité d'organisation comportait G. Bonfante, A. Cichon et C. Pilière.

Jean-Yves Marion organise un séminaire sur les modèles de calcul et la complexité, dans lequel sont impliqués des membres d'autres équipes du LORIA.

8.2 Enseignement

Enseignement en Maitrise et DEA Adam Cichon a enseigné le module de techniques avancées « Logique 2 » en DEA.

Adam Cichon était responsable du DESS Informatique de Nancy jusqu'en Septembre 1999.

Philippe de Groote intervient dans le module de techniques avancées « Démonstration assistée par ordinateur » en DEA. Il assure également un cours de sémantique des langages de programmation à l'École des Mines de Nancy.

Alain Lecomte assure le cours de maîtrise MASS, option « Industrie de la Langue », à l'UPMF de Grenoble.

Encadrement Doctoral Adam Cichon encadre la thèse de Guillaume Bonfante, en collaboration avec François Lamarche et Jean-Yves Marion.

Philippe de Groote encadre la thèse de Catherine Pilière.

Alain Lecomte encadre la thèse de Sylvain Pogodalla.

Jean-Yves Marion encadre la thèse de Jérôme Besombes.

8.3 Jurys de thèse

Adam Cichon a été président du jury de la thèse de Christelle Scharff, UHP, Septembre 1999. François Lamarche a été rapporteur interne de la thèse de Christelle Scharff.

Philippe de Groote et François Lamarche ont été rapporteurs et membres du jury de la thèse de Georges Mounier, Université de Savoie, Février 1999.

François Lamarche a été membre du jury de la thèse de Akim Demaille, Ecole Nationale Supérieure des Communications, Paris, Juin 1999.

8.4 Participation à des colloques, séminaires, invitations

Adam Cichon a participé au quatrième Workshop international «workshop on termination» (WST'99) à Dagstuhl, 10-12 mai 1999.

Jérôme Besombes, François Lamarche et Guy Perrier ont participé au worhshop "Theory, practice and applications of dominance constraints" qui était organisé les 15 et 16 novembre 1999 à l'université de Sarrebrück dans le cadre de l'action Procope "Gemola". François Lamarche et Guy Perrier y ont présenté des communications.

Guillaume Bonfante a présenté une communication intitulée «Two feasible complexity classes within KBO» au quatrième Workshop international «workshop on termination» (WST'99) à Dagstuhl, 10-12 mai 1999.

Philippe de Groote a participé à la conférence Tableaux'99, à Saratoga, du 8 au 11 juin 1999, et aux conférences RTA'99 et CADE'99, à Trente, du 30 juin au 10 juillet 1999. Il a présenté une communication à chacune d'elles.

Philippe de Groote et François Lamarche ont participé aux réunions du réseau TMR sur la logique linéaire à Frascati, du 10 au 12 avril 1999, et à Edimbourg, du 4 au 6 juin 1999. Ils ont également participé à une réunion du projet van Gogh à Utrecht, du 21 au 23 avril, et à un séminaire sur les applications de la logique linéaire, à Dagstuhl, du 22 au 27 août 1999.

Alain Lecomte a donné en 1999, plusieurs exposés au centre de recherches de Xerox. Il a donné une communication au colloque sur la notion de catégorie en logique organisé à l'Université de Neuchâtel (Suisse) en novembre 1998, une communication au colloque MOL6 (Mathematics of Language) qui se tenait à Orlando (Floride) du 23 au 25 juillet, une communication au workshop RLMG dans le cadre de l'Ecole d'été ESSLLI99 à Utrecht, en aout et une communication au Troisiéme Colloque "Logique, Langage et Information" de Tbilissi qui, cette année se déroulait à Batumi (Géorgie), en septembre. Alain Lecomte a donné un exposé de séminaire à l'IRISA (Rennes) en mars 99. Sylvain Pogodalla a donné de nombreux exposés portant sur l'état d'avancement de sa thèse au centre de recherches de Xerox, il a présenté une partie de ses travaux à LACL98 et une autre au colloque GAT'99 (colloque sur la génŕation automatique de textes) qui se déroulait aussi à Grenoble, en septembre.

Catherine Pilière a présenté un résultat de *computational adequacy* lors du worshop du GDR «Logique classique et programmation» qui s'est tenu à Chambéry les 20 et 21 mai 1999.

Catherine Pilière a présenté une communication à la conférence «FCT'99» qui s'est tenue à Iaşi, Roumanie, du 30 août au 3 septembre 1999.

Jean-Yves Marion a présenté *Termination proofs and computational complexity classes*, au quatrième Workshop international «Workshop on Termination» (WST'99), Dagstuhl, 10-12 Mai 1999.

Jean-Yves Marion a présenté *Complexité des algorithmes définis par des systèmes de réécriture avec interprétation polynomiale.*, LIAFA, GDR-ALP, 2 Fev. 99, organisé par I. Guessarian.

Jean-Yves Marion a été invité à présenter *Ordre de terminaison et Complexité*, Journées d'informatique messine (JIM'99), 18-19 Mai, organisées par M. Margenstern.

Jean-Yves Marion a présenté *Feasible complexity and syntactic termination order: two is better than one*, dans le workshop satellite de FLOC 99, «Implicit Computational Complexity », 30 Juin, Trente, Italie.

Sylvain Pogodalla a presenté ses travaux sur la génération à GAT'99 (Génération Automatique de Texte) à Grenoble, 30 septembre 1er octobre.

9 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, « Algorithms with Polynomial Interpretation Termination Proof », *Journal of functional programming*, septembre 1999, à paraître.

[2] A. Cichon, E. Tahhan-Bittar, « Ordinal recursive bounds for Higman's theorem », *Theoretical Computer Science 201*, 1-2, juillet 1998, p. 63–84.

[3] P. de Groote, « An algebraic correctness criterion for intuitionistic multiplicative proof-nets », *Theoretical Computer Science*, 1999.

[4] A. Lecomte, C. Retoré, « Words as modules : a lexicalised grammar in the framework of linear logic proof nets. », *in : Mathematical and Computational Analysis of Natural Language (Proceedings of International conference on mathematical linguistics II), Tarragone*, C. Martin-Vide (éditeur), *45*, John Benjamins, p. 129–144, juin 1998.

[5] D. Leivant, J.-Y. Marion, « Ramified Recurrence and Computational Complexity IV : Predicative Functionals and Poly-Space », *Information and Computation*, 1998, à paraître.

[6] G. Perrier, « Labelled Proof Nets for the Syntax and Semantics of Natural Languages », *Logic Journal of the IGPL 7*, 5, September 1999, p. 629–654.

[7] G. Perrier, « A PSPACE fragment of Second Order Linear Logic », *Theoretical Computer Science 224*, 1-2, 1999, p. 267–289.

Livres et monographies

[8] A. Lecomte, F. Lamarche, G. Perrier (éditeurs), *Logical Aspects of Computational Linguistics - selected papers from LACL'97, Nancy, LNCS, 1582*, Springer-Verlag, 1999.

Articles et chapitres de livre

[9] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, « Algorithms with Polynomial Interpretation Termination Proof », *Journal of functional programming*, septembre 1999, à paraître.

[10] P. de Groote, « An algebraic correctness criterion for intuitionistic multiplicative proof-nets », *Theoretical Computer Science 224*, 1999, p. 115–134.

[11] D. Leivant, J.-Y. Marion, « A characterization of alternating log time by ramified recurrence », *Theoretical Computer Science*, 1999, à paraître.

[12] J.-Y. Marion, « From multiple sequent for Additive Linear Logic to decision procedures for Free Lattices », *Theoretical Computer Science 224*, 1-2, 1999, p. 157–172.

[13] G. Perrier, « Labelled Proof Nets for the Syntax and Semantics of Natural Languages », *Logic Journal of the IGPL 7*, 5, September 1999, p. 629–654.

[14] G. Perrier, « A PSPACE fragment of Second Order Linear Logic », *Theoretical Computer Science 224*, 1-2, 1999, p. 267–289.

Communications à des congrès, colloques, etc.

[15] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, « Complexity classes and rewrite systems with polynomial interpretation », *in : CSL'98, Brno, République Tchèque*, G. Gottlob, G. Etienne, S. katrin (éditeurs), *Lecture Notes in Computer Science, 1584*, Springer, p. 372–384, août 1999.

[16] A. Cichon, E. Tahhan-Bittar, « Strictly Orthogonal Left Linear Rewrite Systems and Primitive Recursion », *in : Proceedings of the XI Simposio Latinoamericano de Lógica.* , *Rapporto di Ricerca del Dipartimento de Filosifia*, 1999. A paraître dans un numéro spécial de la revue Annals of Pure and Applied Logic.

[17] P. de Groote, « A dynamic programming approach to categorial deduction », *in : 16th International Conference on Automated Deduction*, H. Ganzinger (éditeur), *Lecture Notes in Artificial Intelligence, 1632*, p. 1–15, 1999.

[18] P. de Groote, « On the Strong Normalisation of Natural Deduction with Permutation-Conversions », *in : 10th International Conference on Rewriting Techniques and Applications, RTA'99*, P. Narendran, M. Rusinowitch (éditeurs), *Lecture Notes in Computer Science, 1631*, p. 45–59, 1999.

[19] P. de Groote, « The Non-associative Lambek calculus with product in polynomial time », *in : Automatic Reasoning with Analytic Tableaux and Related Methods*, N. V. Murray (éditeur), *Lecture Notes in Artificial Intelligence, 1617*, p. 128–139, 1999.

[20] A. Lecomte, C. Retoré, « Towards a minimal logic for minimalism », *in : Formal Grammar'99*, G. van Kruijf (éditeur), ESSLLI'99, Utrecht, 1999.

[21] A. Lecomte, « Multimodal Categorial Grammars and Minimalist Grammars: a comparaison between two approaches », *in : Proceedings of the Workshop Ressource Logics and Minimalist Grammars*, C. Retoré, E. Stabler (éditeurs), ESSLLI99, Utrecht, 1999.

[22] A. Lecomte, « Partial proof-nets, hybrid logics and minimalist representations », *in : Proceedings of the sixth Conference on Mathematics of Language*, J. Rogers (éditeur), University of Central Florida, Orlando, 1999.

[23] C. Pilière, « A Confluence Result for a Typed lambda-Calculus of Exception Handling with Fixed-point », *in : FCT'99, Iasi, Romania*, G. Ciobanu, G. Paun (éditeurs), *Lecture Notes in Computer Science, 1684*, p. 421–432, août 1999.

Rapports de recherche et publications internes

[24] G. Bonfante, F. Lamarche, « Constructing orders by means of inductive definitions », *Rapport de recherche*, LORIA, mai 1999.

[25] E. Cichon, J.-Y. Marion, « The Light Lexicographic path Ordering », *rapport de recherche n° 99-R-138*, LORIA, Nancy, France, septembre 1999, soumis à Applicable Algebra in Engineering Communication and Computing.

[26] S. Grigorieff, J.-Y. Marion, « Kolmogorov complexity and non-determinism », *rapport de recherche n° 99-R-141*, LORIA, Nancy, France, octobre 1999, soumis à Theoretical Computer Science.

[27] J.-Y. Marion, « Light Multiset path ordering and Ptime - Two is better than one », *rapport de recherche n° 99-R-106*, LORIA, Nancy, France, mars 1999, Présenté à ICC99, Trente, soumis à Information and Computation.

[28] G. Perrier, « From Intuitionistic Proof Nets to Interaction Grammars », *rapport de recherche n° 99-R-120*, LORIA, Nancy, France, 1999.