INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet CALLIGRAMME

Logique Linéaire, Réseaux de Démonstration et Grammaires Catégorielles

Lorraine

THÈME 2A

Table des matières

1 Composition de l'équipe

Responsable scientifique

Philippe de Groote [CR INRIA jusqu'au 31.08.01, DR INRIA à partir du 01.09.01]

Responsable permanent

François Lamarche [DR INRIA]

Assistante de projet

Geneviève Pierrelée Grisvard [jusqu'au 31.03.01]

Céline Simon [du 01.04.01 au 31.08.01]

Hélène Zganic [à partir du 01.09.01]

Personnel INRIA

Bruno Guillaume [CR, à partir du 01.09.01]

Personnel Universitaire

Adam Cichon [Professeur, Université Henri Poincaré]

Jean-Yves Marion [Maître de conférences, Université Nancy 2, détaché à l'INRIA jusqu'au 31.08.01]

Guy Perrier [Maître de conférences, Université Nancy 2, détaché au CNRS à partir du 01.09.01]

Guillaume Bonfante [Maître de conférences, Ecole des Mines de Nancy, à partir du 01.09.01]

Chercheurs doctorants

Jérôme Besombes [allocataire MESR, Université Henri Poincaré, Moniteur à l'Université de Metz]

Paulin Jacobé de Naurois [étudiant en quatrième année à l'ENS Lyon, à partir du 01.10.01, thèse coencadrée par les projets Protheo et Calligramme]

Jean-Yves Moyen [étudiant en quatrième année à l'ENS Lyon jusqu'au 31.08.01, allocatire normalien à partir du 01.09.01]

Catherine Pilière [Université Henri Poincaré, ATER à l'IUT de Metz, jusqu'au 31.08.01]

Sylvain Pogodalla [boursier CIFRE, Institut National Polytechnique de Lorraine, jusqu'au 31.08.01]

Sylvain Salvati [allocataire MESR, à partir du 01.09.01]

2 Présentation et objectifs généraux

Mots clés : logique linéaire, calcul des séquents, réseau de démonstration, grammaire catégorielle, analyse syntaxique des langues naturelles, sémantique des langues naturelles, lambda-calcul, théorie des types, complexité implicite.

Le projet Calligramme a pour objectif le développement d'outils et de méthodes issus de la théorie de la démonstration et, en particulier, de la logique linéaire. Deux champs d'application sont privilégiés : dans le domaine de la linguistique computationnelle, la modélisation logique de la syntaxe et de la sémantique des langues naturelles ; dans le domaine du génie logiciel, l'étude de la terminaison et de la complexité des programmes.

3 Fondements scientifiques

Les recherches du projet Calligramme se situent à l'intersection de la logique mathématique et de l'informatique. Les domaine scientifiques sur lequels nous nous appuions sont la théorie de la démonstration, le λ-calcul et, plus particulièrement, la logique linéaire. Cette dernière, due à J.-Y. Girard [Gir87], résulte 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 :

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

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. 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.

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 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

[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		AB	$A \oplus B$	
implication		$A \multimap B$		
constantes		$\mathbf{1}, \perp$	$\top, \mathbf{0}$	
modalités				$!A, ?A$

Table 1 : *Les connecteurs de la logique linéaire*

est réalisé grâce à deux modalités permettant un usage contrôlé des règles structurelles. 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 régies par l'utilisation de nouveaux connecteurs. C'est cette idée originale qui confère à la logique linéaire toute sa finesse et sa puissance.

Les activités du projet Calligramme s'organisent autour de trois actions de recherche :
- réseaux de démonstration, calcul des séquents et λ-calculs typés ;
- grammaires catégorielles ;
- complexité implicite des calculs.

La première de ces actions est essentiellement théorique. Les deux autres, qui présentent à la fois un caractère théorique et appliqué, correspondent à nos deux champs d'application privilégiés.

3.1 Réseaux de démonstration, calcul des séquents et lambda-calculs typés

Mots clés : calcul des séquents, réseau de démonstration, lambda-calcul, isomorphisme de Curry-Howard, théorie des types, sémantique dénotationnelle..

Participants : Guillaume Bonfante, Philippe de Groote, François Lamarche, Guy Perrier, Catherine Pilière.

Résumé : *Le but de cette action est de développer les outils théoriques que nous employons dans nos autres actions de recherche. Nous nous intéressons, en particulier, à la notion même de démonstration formelle, tant d'un point de vue syntaxique (dérivation séquentielles, réseaux de démonstration, λ-termes) que d'un point de vue sémantique.*

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

Les réseaux de démonstrations sont des représentations graphiques (au sens de la théorie des graphes) des démonstrations 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 ;
- 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.

En fait, jusqu'à l'invention des réseaux de démonstration, l'outil principal de représentation des démonstrations dans les logiques constructives était le lambda-calcul. Ceci est dû à l'isomorphisme de Curry-Howard, qui établit une correspondance entre systèmes de déduction narturelles pour les logiques intuitionistes et λ-calcul typés.

Bien que l'isomorphisme de Curry-Howard doivent son existence au caractère fonctionnel de la logique intuitionniste, il 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 et ses applications restent présents comme thème de recherche au sein du projet.

3.2 Grammaires catégorielles

Mots clés : grammaire catégorielle, sémantique de Montague, inférence grammaticale, analyse syntaxique des langues naturelles, sémantique des langues naturelles.

Participants : Jérôme Besombes, Guillaume Bonfante, Philippe de Groote, Bruno Guillaume, Jean-Yves Marion, Guy Perrier, Sylvain Pogodalla, Sylvain Salvati, François Lamarche.

[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.

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.*

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 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 :

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

[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.

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.

3.3 Complexité implicite des calculs

Mots clés : théorie de la complexité, théorie de la programmation, types, lambda-calcul, isomorphisme de Curry-Howard, ordre de terminaison.

Participants : Guillaume Bonfante, Adam Cichon, Paulin Jacobé de Naurois, Jean-Yves Marion, Jean-Yves Moyen.

Résumé : *La construction de logiciels sûrs est une nécessité. Il est crucial dans le développement d'un logiciel certifié de s'assurer de la qualité de l'implantation en terme d'efficacité, et de ressources de calcul. La complexité implicite est une approche à l'analyse des ressources employées par un programme. Pour cela, les outils proviennent essentiellement de la théorie de la démonstration. L'objectif est de compiler un programme en certifiant sa complexité.*

La méta-théorie de la programmation répond traditionnellement à des questions de correction par rapport à une spécification, comme la terminaison. Ces propriétés sont dites extensionnelles. Cependant, certaines propriétés, comme l'efficacité d'un programme et les ressources employées pour effectuer un calcul, sont exclues de cette méthodologie. La cause de cette lacune tient à la nature des questions posées. Dans le premier cas, nous traitons une propriété extensionnelle, tandis que dans le second cas, nous abordons la question sur la manière dont la fonction est réalisée et comment un calcul est effectué. Dès lors, nous nous intéressons à une propriété intensionnelle des programmes. Pourtant, la maîtrise de ces facteurs permet de s'assurer de la qualité d'une implantation.

La complexité d'un programme est une mesure des ressources nécessaires à son exécution. Les ressources qui sont prises en compte sont, usuellement, le temps et l'espace. La théorie de la complexité étudie les problèmes et les fonctions qui sont calculables avec une certaine quantité de ressources. Il ne faut pas confondre la complexité d'une fonction avec la complexité d'un programme. Une fonction est réalisée par différents programmes. Certains programmes sont efficaces, d'autres ne le sont pas.

Un succès de la théorie de la complexité est de préciser à « l'expert en programmation » les limites de son art, et ceci quels que soient les giga-octets et les méga-flops à sa disposition. Un autre succès de la théorie de la complexité est de fournir un modèle mathématique de la complexité algorithmique. Mais face à ces modèles, l'expert en programmation est en plein désarroi. Les causes en sont diverses, illustrons-les par deux exemples. Le théorème d'accélération linéaire affirme que tout programme qui s'exécute en temps $T(n)$ (où n est la taille de l'entrée) peut être transformé en un programme équivalent qui calcule en temps $\epsilon T(n)$ où ϵ est aussi petit que nous voulons. Ce résultat n'a aucune contrepartie *réelle*. Par ailleurs, une

fonction est faisable si elle est calculée par un programme dont la complexité est acceptable. La classe des fonctions faisables est souvent identifiée avec la classe PTIME des fonctions calculables en temps polynomial. Un résultat typique est de définir un langage de programmation L et de démontrer que la classe des fonctions calculées par les programmes de L, est exactement la classe PTIME. Ce type de résultat ne répond pas aux questions de l'expert en programmation, car le langage de programmation L ne contient pas les « bons algorithmes », qu'il utilise quotidiennement. Le fossé entre les deux disciplines s'explique encore par une différence de point de vue. La théorie de la complexité, fille de la théorie de la calculabilité, a gardé un point de vue extensionnel, dans la modélisation, tandis que la théorie de la programmation est intrinsèquement intensionnelle.

La nécessité de raisonner sur les programmes est une question pertinente dans le processus de développement des logiciels. La certification d'un programme est une propriété essentielle, mais elle n'est pas la seule. Démontrer la terminaison d'un programme de complexité exponentielle n'a pas de sens par rapport à notre réalité. Il se pose alors le problème de la construction d'outils pour raisonner sur les algorithmes. La complexité implicite des calculs essaie de faire face à ce vaste chantier, qui consiste à analyser la complexité des algorithmes.

4 Domaines d'applications

4.1 Modélisation de la syntaxe et de la sémantique des langues naturelles

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.

4.2 Terminaison et complexité des programmes

La thèorie de la complexité implicite est récente et un long chemin reste encore à parcourir. De ce fait, le transfert des outils théoriques actuels vers des applications ciblées est important, car il permet de valider et guider nos hypothèses. Pour cela, trois directions ont été prises.

1. Programmation fonctionnelle du premier ordre. Un premier prototype, ICAR, a été développé qui devrait être intégré à ELAN.
2. Extraction de programme efficace à partir de démonstration. Il s'agit de construire des théories logiques dans lesquels les programmes extraits, par l'isomorphisme de Cury-Howard, sont efficaces.

3. Application aux systèmes avec codes mobiles. Ce travail vient de débuter en collaboration avec les projets Crystal et Mimosa.

5 Logiciels

Dans leurs états actuels, les logiciels développés au sein du projet Calligramme ont un statut de prototypes nous permettant de valider nos idées.

- AGIR est un analyseur syntaxique pour le français, fondé sur le formalisme des Grammaires d'Interaction et utilisant la résolution de contraintes. Ce prototype a été écrit en Oz[Smo95], un langage qui fait la synthèse de plusieurs paradigmes et permet, notamment, la programmation par contraintes. L'objectif de l'implémentation est à la fois d'éprouver la pertinence linguistique du formalisme et de construire et de tester une grammaire modulaire du français à large couverture
- RDTLL est un système d'apprentissage des grammaires de dépendances, développé en java. Le programme lit un corpus de phrases étiquetées par des dépendances (liens orientés entre les mots) et produit une grammaire de dépendances. Il s'agit d'un apprentissage par exemples positifs : le programme lit un nombre fini d'éléments d'un langage inconnu et produit à partir de ces exemples une grammaire compatible. A court terme, l'évolution de ce prototype devra permettre la prise en compte des entrées/sorties dans un format standard (XML) ainsi que la gestion de la représentation graphique des ces éléments à l'écran.
- ICAR est un logiciel qui permet d'analyser la complexité implicite de certains systèmes de réécriture. Il est décrit plus amplement dans [17]. ICAR fonctionne en deux étapes. Tout d'abord, le système de réécriture est lu, et un ordre (MPO ou LPO) par lequel le système termine est recherché. Ensuite, une quasi-interprétation des symboles est lue et le système vérifie qu'elle est compatible. En fonction des résultats, ICAR peut être capable de dire si la fonction spécifiée par le système de réécriture est dans Ptime ou Pspace. ICAR a été écrit en Objective Caml, il a été ensuite réécrit en TOM pour pouvoir être utilisé directement avec le système Elan développé au sein du projet PROTHEO.

6 Résultats nouveaux

6.1 Réseaux de démonstration, calcul des séquents et lambda-calculs typés

6.1.1 Réseaux de démonstration intuitionnistes

Partant des travaux de François Lamarche[Lam94], Guy Perrier a conçu une représentation abstraite des réseaux réseaux de démonstration multiplicatifs intuitionnistes [18].

[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.

[Lam94] F. Lamarche, « Proof Nets for Intuitionistic Linear Logic I: Essential Nets », *Preliminary report*, Imperial College, avril 1994.

Le caractère orienté d'un réseau intuitionniste induit un ordre entre les formules atomiques qu'il contient. Cet ordre concentre l'essence du réseau si bien que l'on peut oublier tout le reste. Un réseau apparaît alors comme un graphe acyclique orienté dont les sommets sont les formules atomiques qu'il contient et dont les arcs indiquent qu'une formule atomique précède immédiatement une autre dans le réseau. Si l'on se restreint au fragment implicatif de la logique linéaire intuitionniste (celui qui comporte l'implication linéaire comme unique connecteur), ce graphe se réduit à un arbre.

Sous cet angle nouveau, démontrer un théorème revient à chercher un graphe acyclique orienté vérifiant une spécification donnée et répondant à certains critères particuliers. La spécification exprime sous forme d'un graphe acyclique orienté partiellement spécifié le théorème à démontrer.

Une première application de ce résultat est l'implémentation de la démonstration de théorèmes en logique linéaire intuitionniste multiplicative comme un problème de satisfaction de contraintes. Cette application a été largement inspirée par les travaux de Denys Duchier sur les contraintes de domination[DT99].

Une seconde application touche en linguistique à la modélisation de la syntaxe des langues (voir paragraphe 6.2.3)

6.1.2 Structades

Depuis quelques années on a assisté, notamment en linguistique computationelle[Moo97, Mor94], à une grande prolifération de versions non-commutatives de la logique linéaire. Une question naturelle est de savoir si ces différents formalismes peuvent être vus comme les instances d'un cadre général.

Une telle théorie logique générale nécessite avant tout une théorie générale des contextes : étant donné une logique, on doit pouvoir définir la structure qui « lie » les formules dans un séquent donné. Vient ensuite la question de savoir quels sont les différentes logiques qui possèdent la même théorie des contextes.

La notion de structade, développée par François Lamarche [20], répond à ces questions de façon détaillée, dans un cadre purement multiplicatif. L'observation de base est le fait que dans une théorie algébrique linéaire (c'est à dire un ensemble Σ de symboles fonctionnels et un ensemble d'équations entre les termes construits sur Σ, avec la condition qu'exactement les mêmes variables apparaissent une et une seule fois à gauche et à droite d'une équation) on peut ajouter aux termes une variable supplémentaire à la sortie, et donc effacer la distinction entre les entrées (arguments) d'un terme et la sortie. De plus, en considérant des ensembles d'inéquations (au lieu d'équations), on peut prendre en compte des règles d'entropie (appelées également règles de réécriture structurelles, dans le cadre des grammaires multimodales[Moo97]).

[DT99] D. Duchier, S. Thater, « Parsing with Tree Descriptions: a constraint based approach », *in: Natural Language Understanding and Logic Programming NLULP'99,Dec 1999, Las Cruces, New Mexico*, 1999.

[Moo97] M. Moortgat, « Categorial Type Logic », *in: Handbook of Logic and Language*, J. van Benthem et A. ter Meulen (éditeurs), Elsevier, 1997, ch. 2.

[Mor94] G. Morrill, *Type Logical Grammar: Categorial Logic of Signs*, Kluwer Academic Publishers, Dordrecht, 1994.

Le travail, qui fait une correspondance détaillée entre la présentation syntaxique et le formalisme algébrique, se termine par un théorème d'élimination des coupures totalement générique, dans lequel *tous* les théorèmes délimination des formalismes sous-structurels peuvent se retrouver comme des cas particuliers.

6.1.3 Logique linéaire et produits dépendants

Les travaux de F. Pfenning[CP96] permettent d'intégrer syntaxiquement la logique linéaire aux types dépendants grâce à un calcul des séquents où les contextes sont en deux parties : une partie intuitionniste (où se fait en particulier la gestion des types dépendants) et une partie linéaire. D'un point de vue sémantique, suite à une suggestion de M. Hoffmann et Th. Streicher, on connaissait les propriétés catégoriques que doit posséder tout modèle d'un tel calcul. Néanmoins, aucun modèle concret n'avait été développé.

G. Bonfante et F. Lamarche ont résolu ce problème [19]. Le modèle qu'ils proposent est construit à parir de la catégorie des (multi)graphes et s'exprime en termes d'une fibration de Grothendieck. Chaque fibre est un modèle de la logique linéaire intuitionniste, en d'autres termes une catégorie symétrique monoïdale fermée. En fait cette structure symétrique monoïdale est une généralisation de celle employée dans la thèse de G. Bonfante pour obtenir des définitions inductives de structures ordonnées[Bon00].

6.1.4 Contenu algorithmique de la logique classique

Dans sa thèse [6], Catherine Pilière a étudié un λ-calcul doté d'un mécanisme de traitement d'exceptions. La version simplement typée de ce calcul, introduite par Philippe de Groote[dG95], a pour système de typage la logique propositionnelle classique. Afin d'étudier les propriétés de ce calcul dans un cadre de programmation réaliste, Catherine Pilière y a ajouté un opérateur de point fixe autorisant la récursion générale. Ce calcul enrichi est présenté sous trois formes sémantiques : respectivement un système de règles de réduction, une sémantique opérationnelle et un modèle dénotationnel basé sur la notion de transformation par continuations. L'équivalence de ces trois interprétations est établie sur une classe particulière de termes (les programmes), montrant ainsi en quoi l'approche contrôlée du traitement d'exceptions peut être considérée dans certains cas comme une alternative raisonnable au traitement classique.

Philippe de Groote a défini une extension du $\lambda\mu$-calcul de Parigot[Par92], où la disjonction intuitionniste est prise comme primitive [13]. La relation de réduction qui lui est associée

[CP96] I. Cervesato, F. Pfenning, « A Linear Logical Framework », *in: Proceedings of the Eleventh Annual Symposium on Logic in Computer Science — LICS'96*, E. Clarke (éditeur), IEEE Computer Society Press, p. 264–275, 1996.

[Bon00] G. Bonfante, *Constructions d'ordres, analyse de la complexité*, Thèse de doctorat, Institut National Polytechnique de Lorraine, 2000.

[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.

[Par92] M. Parigot, « $\lambda\mu$-Calculus: an Algorithmic Interpretation of Classical Natural Deduction », *in: Proceedings of the International Conference on Logic Programming and Automated Reasoning*, A. Voronkov (éditeur), *624*, Springer Verlag, p. 190–201, 1992.

comprend les conversions permutatives. Elle est confluente et fortement normalisable, ce qui permet de prouver que le calcul satisfait la propriété de la sous-formule.

6.2 Grammaires catégorielles

6.2.1 Grammaires catégorielles stochastiques

Afin de permettre, à terme, un traitement catégoriel de la langue qui soit robuste, efficace et en taille réelle, Guillaume Bonfante et Philippe de Groote ont développé un modèle stochastique de grammaire catégorielle[11].

Le point de départ de ce travail est que la recherche de démonstration dans le calcul de Lambek peut être exprimée à l'aide d'un automate à pile non déterministe. Les deux mouvements, (empiler et dépiler) correspondent à lier un atome d'une catégorie syntaxique à gauche ou à droite. Ceci permet d'associer à tout atome la probabilité qu'il soit lié à gauche ou à droite.

Le modèle de grammaire stochastique qui en résulte présente, a priori, d'intéressantes propriétés. Les probabilités sont attachées aux dépendances syntaxiques et non à des règles de production. Elles sont donc exprimées au niveau du lexique, ce qui permet une lexicalisation totale du formalisme. De plus, la désambiguisation lexicale tient compte des dépendances à longue distance.

6.2.2 Grammaires catégorielles abstraites

Les grammaires catégorielles traditionnelles ne permettent pas de rendre compte de nombreux phénomènes syntaxiques présents dans les langues naturelles. Ceci a donné lieu, dans la littérature, à de nombreuses extensions. Malheureusement, les fondements mathématiques de ces extensions ne sont pas toujours bien établis, ce qui les rend difficile à implanter.

Afin de pallier ces inconvénients, Philippe de Groote à développer une notion de grammaire catégorielle abstraite, qui présente l'avantage d'être basée sur un petit nombre de primitives mathématiques clairement identifiées [14]. De plus, cet ensemble de primitives permet de traiter à la fois syntaxe et sémantique. En conséquence, le formalisme est réversible.

Philippe de Groote, Bruno Guillaume et Sylvain Salvati étudient les propriétés de ces grammaires catégorielles abstraites. Du point de vue des langages formels générés, elles sont strictement plus expressives que les grammaires catégorielles traditionnelles. En outre, elles permettent de représenter, de manière structurellement équivalente, de nombreux autres formalismes utilisés en linguistique computationnelle (automates à états finis, transducteurs rationnels, grammaires hors-contexte, automates et transducteurs d'arbres, grammaires d'arbres adjoints, etc.)

6.2.3 Grammaires d'interaction

Guy Perrier a continué à développer les grammaires d'interaction[Per00], qui est un formalisme grammatical issu de travaux sur les réseaux de démonstration de la logique linéaire

[Per00] G. Perrier, « Interaction Grammars », *in: Proceedings of the 18th International Conference on Computational Linguistics, COLING 2000*, Universität des Saarlandes, p. 600–606, 2000.

intuitionniste.

Dans le fragment implicatif de la logique linéaire intuitionniste les réseaux de démonstrations peuvent être réduit à des arbres représentant un ordre entre formules atomiques [18]. Démontrer un théorème revient alors à chercher un modèle valide d'une description d'arbre.

Comme, dans le paradigme des Grammaires Catégorielles, l'analyse syntaxique des langues se réduit à la démonstration de théorèmes, le résultat précédent permet d'envisager l'analyse syntaxique d'une phrase comme la recherche d'un modèle valide d'une description d'arbre. Cette description d'arbre exprime la fonction syntaxique possible des différents mots de la phrase et le modèle, s'il existe, constitue l'arbre syntaxique de la phrase analysée.

L'intérêt de cette vision de l'analyse syntaxique est qu'elle exprime simplement certains mécanismes linguistiques fondamentaux et qu'elle est suffisamment souple pour pouvoir être enrichie de façon à exprimer la complexité d'une langue.

Actuellement, Guy Perrier se concentre sur les problèmes que pose le passage, pour une langue donnée, d'une grammaire jouet à une grammaire à large couverture. Une des questions clés est celle de la taille de la grammaire, c'est-à-dire celle du lexique étant donné que les Grammaires d'Interaction sont lexicalisées. Une solution consiste à organiser la grammaire en un ensemble de modules liés par une relation d'héritage multiple, chaque module correspondant à une construction grammaticale particulière de la langue. Le lexique est ensuite obtenu par croisement des modules terminaux de la grammaire.

6.2.4 Apprentissage grammatical

L'étude de l'inférence grammaticale a débuté récemment, au sein du projet Calligramme. Les méthodes que nous employons sont symboliques (par opposition à statistiques) et suivent le paradigme d'apprentissage par identification tel que défini dans les travaux de Gold et Angluin.

Notre objectif est de définir des modèles d'apprentissage pour les grammaires catégorielles ou apparentées (notamment, les grammaires minimalistes et les grammaires de dépendances). Dans ce cadre, Jérôme Besombes et J-Y Marion ont étudié, implanté et expérimenté un algorithme d'apprentissage des grammaires de dépendances, dites réversibles [10].

6.2.5 Interface syntaxe/sémantique

Les rapports entre grammaire catégorielle et sémantique de Montague sont très étroits. Il existe, en effet, un homomorphisme entre les termes sémantiques assignés par Montague à chaque constituant d'une phrase, d'une part, et les déductions catégorielles permettant de conclure à la bonne formation syntaxique de ces constituants, d'autre part. Cette observation, due à van Benthem[vB86], est à la base de l'interface entre syntaxe et sémantique que présente les grammaires catégorielles modernes.

Dans sa thèse [7], Sylvain Pogodalla exploite cette interface dans le sens sémantique/syntaxe, c'est-à-dire, dans le sens de la génération. Il montre, en particulier, que le problème de génération est décidable pour le calcul de Lambek et pour des représentations sémantiques linéaires avec une constante au moins. Ce résultat et les algorithmes qui l'accompagnent sont basés sur la théorie des réseaux de démonstration et la géométrie de l'interaction.

[vB86] J. van Benthem, *Essays in Logical Semantics*, Reidel, Dordrecht, 1986.

Les techniques employées par Montague pour obtenir une sémantique compositionnelle pour les langues naturelles présentent des analogies frappantes avec la notion de continuation telle qu'employée en théorie des langages de programmation. Par ailleurs, ces mêmes continuations permettent d'exprimer le contenu algorithmique de la logique classique (voir paragraphe 6.1.4). Cette observation a été exploitée par Philippe de Groote qui montre comment le $\lambda\mu$-calcul de Parigot permet de faciliter l'expression des composantes sémantiques lexicales [15].

6.3 Complexité implicite des calculs

6.3.1 Complexité des programmes fonctionnelles du 1$^{\text{er}}$ ordre

Les méthodes de preuve de terminaison offrent un formidable outil pour analyser la complexité d'un programme fonctionnelle du 1$^{\text{er}}$ordre [8]. A partir d'ordres de terminaison classiques, utilisés en réécriture, nous sommes capables de déterminer la complexité (e.g temps ou espace polynomial) d'un programme, à l'aide d'une analyse prédicative.

Ainsi dans [12], Guillaume Bonfante, Jean-Yves Marion et Jean-Yves Moyen ont démontré que les fonctions calculables en espace polynomial sont exactement les fonctions définies par un programme qui termine par LPO et qui admet une quasi-interprétation polynomiale. Nous travaillons actuellement sur des techniques de terminaison plus puissante pour capturer des classes d'algorithme plus importantes.

Dans le même ordre d'idée, Adam Cichon, en collaboration avec E. Tahhan-Bittar, a étudié la complexité des systèmes de réécriture strictement orthogonaux [9]. Il en résulte une caractérisation syntaxique de la hiérarchie de Grzegorczyk.

6.3.2 Complexité des programmes extraits de démonstrations

La nécessité de raisonner sur les algorithmes est une question des plus pertinentes dans le cadre de l'extraction semi-automatisée d'un programme à partir de la démonstration de la correction d'une spécification. Cette approche est partagée par les systèmes Coq[1] et Nuprl[2]. La théorie sous-jacente assure que le programme réalise correctement ce qui est demandé. Cependant, l'efficacité du programme extrait n'est pas garantie. Ainsi, comme l'argumente R. Constable (créateur de Nuprl), pour discerner les bons algorithmes, ou les bonnes démonstrations ce qui revient au même dans ce contexte, il faut pouvoir raisonner sur les algorithmes.

J-Y Marion a construit une théorie du premier ordre qui caractérise les fonctions calculables en temps polynomial [16]. Il travaille actuellement à une extension de cette théorie logique qui permettrait de raisonner sur les algorithmes.

7 Contrats industriels (nationaux, européens et internationaux)

Le projet Calligramme entretient, depuis sa création, des relations avec le centre de recherche de Xerox à Grenoble. Cette collaboration s'est concrétisée en 1998 par l'inscription en thèse, sur contrat CIFRE, de Sylvain Pogodalla. Sylvain Pogodalla a soutenu sa thèse en 2001

[1] `http://pauillac.inria.fr/coq/coq-eng.html`
[2] `http://www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html`

et effectue actuellement un postdoc dans les centres de recherche de Xerox de Grenoble et de Palo alto.

Guy Perrier a été contacté par Daniela Garcia du service *Modélisation et Technologies de l'Information* de l'unité Recherche et Développement d'EDF de Clamart. Son équipe a développé un éditeur de graphes terminologiques, WorldTrek-édition, qu'elle souhaiterait adapter pour en faire un éditeur de grammaires modulaires. Elle est donc intéressée par les travaux de Guy Perrier sur le sujet. Une première rencontre à la fin du mois de mai à permis de tracer les grandes lignes d'une collaboration qui a commencé à se concrétiser par une première implémentation des Grammaires d'Interaction avec WorldTrek-édition.

8 Actions régionales, nationales et internationales

8.1 Actions régionales

Le projet Calligramme participe à deux thèmes du pôle *Intelligence Logicielle* du Contrat Plan Etat-Région 2000-2006 :

- qualité et sûreté des logiciels et systèmes informatiques ;
- ingénierie de la langue, du document, et de l'information scientifique technique et culturelle.

8.2 Actions nationales

Le projet Calligramme participe à deux Action de Recherche Coopérative :

- GRACQ (Acquisition de Grammaires Catégorielles) dont les autres participants sont les projets Aïda et Paragraphe de l'IRISA (Rennes), le thème TALN de l'IRIN (Nantes), et l'équipe Grappa du LIFL (Lille).
- RLT (Ressources Linguistiques pour les TAGs) dont les autres participants sont le projet Attol de l'INRIA Rocquencourt, le projet Langue et Dialogue du LORIA (Nancy) et l'équipe TALaNa de Paris 7.

8.3 Actions européennes

Calligramme participe au réseau TMR *Linear Logic in Computer Science*. Ce réseau regroupe des les sites suivants : Marseille (Institut de mathématiques de Luminy, projet Calligramme et centre de recherche Xerox de Grenoble — rattachés 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 (PPS — Paris 7, LIPN — Paris 13, ENS) et Rome (Universités de Rome 1, Rome 3, Bari, Sienne, Trente et CNR-IAC).

8.4 Visites et invitations de chercheurs

- Mitch Harris, Doctorant, Université Urbana-Champaign, a effectué un stage de trois mois dans le cadre de l'opération QSL ACT (mai–juillet 2001).

– James Royer, Professeur à l'Université de Syracure, a bénéficié d'une invitation INRIA d'un mois en vue d'une collaboration sur le thème de l'apprentissage grammatical (juillet 2001).

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

– Adam Cichon est membre de la commission de spécialistes de l'UHP (27^{e} section).
– Philippe de Groote est vice président et membre de la commission permanente du comité des projets du LORIA et de l'INRIA-Lorraine, membre de l'équipe de direction du LORIA, membre du Conseil d'Orientation Scientifique du LORIA, membre nommé du Conseil de Laboratoire du LORIA, membre titulaire de la commission de spécialistes de l'INPL (27^{e} section). Il a participé au comité de programme de la conférence RTA'01 (Utrecht, Pays-Bas, mai 2001) et a présidé, avec Glyn Morrill, le comité de programme de la conférence LACL'01 (Le Croisic, France, juin 2001). Il est membre du comité éditorial de la collection *Research Papers in Formal Linguistics and Logic* (éditée par l'Université de Bologne).
– François Lamarche est président du comité des bourses du LORIA et de l'INRIA-Lorraine. A ce titre, il est membre de la commission permanente du comité des projets du LORIA et de l'INRIA-Lorraine. Il est membre du bureau DFD de la section informatique de l'école doctorale IAE+M.
– Jean-Yves Marion est responsable adjoint du thème QSL du pôle intelligence logicielle. Il est également, dans le cadre de ce thème, co-responsable de l'action "efficacité, fiabilité et robustesse des composants", coordonateur des journées de séminaire, et animateur scientifique de l'opération ACT. Jean-Yves Marion est responsable local de l'Action de Recherche Coopérative GRACQ (Apprentissage des grammaires catégorielles). Il est membre nommé de la commission de spécialistes de ENS Lyon (27^{e} Section) depuis février 2001. Il a été membre des comités de programme de ICC'01 (satellite de PADO et MFPS, Aarhus, Danemark, mai 2001) et de Rule'01 (satellite de PLI, Florence, Italie, septembre 2001).
– Guy Perrier est membre du conseil des opérations du thème ILD&ISTC du pôle intelligence logicielle. Il est responsable, pour l'équipe Calligramme, de l'Action de Recherche Concertée RLT (Acquisition et Représentation de Ressources Linguistiques pour les TAGs). Il est membre élu du conseil de laboratoire du LORIA, membre suppléant de la commission de spécialistes de Nancy 2 (27^{e} section) et membre élu du conseil d'administration de l'IUT d'Epinal (jusqu'au 1er septembre 2001).

9.2 Enseignement

– Adam Cichon, en collaboration avec Claude Kirchner, donne le cours *Logique et démonstration automatique* du DEA d'informatique des universités nanceiennes.
– Philippe de Groote, en collaboration avec Didier Galmiche, donne le cours *Calculs pour la modélisation et la preuve* du DEA d'informatique des universités nanceiennes. Il inter-

vient dans le cours *Sémantique, logique et pragmatique des langues naturelles* de ce même DEA. Il a également donné un cours intitulé *Introduction to Typed Lambda Calculus and Proof Theory* dans le cadre de l'*International Masters Programme in Computational Logic* de l'Université de Dresde.
- Jean-Yves Marion, en collaboration avec Maurice Margenstern, donne le cours *Modèles de calcul, analyse d'algorithmes et complexité de problèmes* du DEA d'informatique des universités nanceiennes.
- Guy Perrier, en collaboration avec Bertrand Gaiffe, donne le cours *Algorithmique du traitement structurel des langues* du DEA d'informatique des universités nanceiennes.

9.3 Jurys de thèse

- Philippe de Groote a été rapporteur et membre des jurys de thèses de Virgile Mogbil (Université de la Méditerranée, Marseille, janvier 2001) et Sylvain Soliman (Paris VII, avril 2001). Il a été membre du jury de thèse de Catherine Pilière (Université Henri Poincaré, Nancy, décembre 2001).
- François Lamarche a été membre du jury de thèse de Quintijn Puite (Université d'Utrecht, Pays-Bas, janvier 2001).

9.4 Participation à des colloques, séminaires, invitations

- Jérôme Besombe et Jean-Yves Marion ont participé à la première réunion de l'action GRACQ (Paris, mars 2001). Ils ont présenté une communication [**?**] au *third workshop on Learning Language in Logic* (Strasbourg, France, septembre 2001).
- Jérôme Besombes, Guillaume Bonfante, Philippe de Groote, François Lamarche et Jean-Yves Marion ont participé aux journées de rencontre de l'action GRACQ (Nantes, juin-juillet 2001),
- Jérôme Besombes, Guillaume Bonfante, Philippe de Groote, François Lamarche, Jean-Yves Marion et Guy Perrier ont assisté à la conférence internationale LACL'2001 (Le Croisic, France, juin 2001).
- Guillaume Bonfante a participé à *Formal Grammar 01*, (Helsinki, Finlande, aout 2001) où il a présenté un travail réalisé en collaboration avec Philippe de Groote [11].
- Philippe de Groote a présenté ses travaux [13, 14, 15] à TLCA'01 (Cracovie, Pologne, mai 2001), à ACL 2001 (Toulouse, France, juillet 2001) et au *13th Amsterdam Colloquium* (Amsterdam, Pays-Bas, décembre 2001). Il a donné, en collaboration avec Glyn Morrill, le cours *Introduction to Type-Logical Syntax and Semantics* de l'école d'été ESSLLI'01 (Helsinki, Finlande, aout 2001).
- Philippe de Groote et François Lamarche ont participé à la réunion annuelle du réseau européen TMR *Linear Logic in Computer Science* (Bertinoro, Italie, avril 2001) et au *V Roma Workshop on New Perspectives in Logic and Formal Linguistics* (Rome, Italie, mai 2001).
- François Lamarche a participé au séminaire *Computational Linguistics and Logic* (Université d'Utrecht, Pays-Bas, janvier 2001). Il a été invité à donner un exposé dans le cadre du séminaire de linguistique informatique de l'Université de Nantes (février 2001).

Il a participé au séminaire de catégories du Chevaleret (Paris, octobre et décembre 2001), au séminaire itinérant de catégories (Amiens, novembre 2001).
- Jean-Yves Marion a présenté une communication [16] à CSL 2001 (Paris, France, septembre 2001).
- Jean-Yves Moyen s'est rendu à la *Andrei Ershov Fourth International Conference* (Novosibirsk, Russie, juin-juillet 2001) afin d'y présenté un travail réalisé en collaboration avec Guillaume Bonfante et Jean-Yves Marion [12]. Il a présenté le logiciel ICAR à ICC'01 (Aarhus, Danemark, mai 2001) et à Rule'01 (Florence, Italie, septembre 2001) [17].
- Guy Perrier a présenté une communication [18] à LPAR 2001 (Havana, Cuba, décembre 2001).

10 Bibliographie

Ouvrages et articles de référence de l'équipe

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

[2] R. David, B. Guillaume, « A λ-calculus with explicit weakening and explicit substitution », *Mathematical Structures in Computer Science 11*, 1, 2001, p. 169–206.

[3] P. de Groote, « An algebraic correctness criterion for intuitionistic multiplicative proof-nets », *Theoretical Computer Science 224*, 1–2, 1999, p. 115–134.

[4] D. Leivant, J.-Y. Marion, « A characterization of alternating log time by ramified recurrence », *Theoretical Computer Science 236*, 1-2, 2000, p. 192–208.

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

Thèses et habilitations à diriger des recherches

[6] C. Pilière, *Une approche contrôlée du traitement d'exceptions en programmation fonctionnelle*, Thèse de doctorat, Université Henri Poincaré, 2001.

[7] S. Pogodalla, *Réseaux de preuve et génération pour les grammaires de types logiques*, Thèse de doctorat, Institut National Polytechnique de Lorraine, 2001.

Articles et chapitres de livre

[8] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, « Algorithms with polynomial interpretation termination proof », *Journal of Functional Programming 11*, 1, 2001, p. 33–53.

[9] A. Cichon, « Strictly orthogonal left linear rewrite systems and primitive recursion », *Annals of Pure and Applied Logic 11*, 1–3, 2001, p. 79–101.

Communications à des congrès, colloques, etc.

[10] J. Besombes, J.-Y. Marion, « Identification of reversible dependency tree languages », *in : third workshop on Learning Language in Logic (LLL 01)*, L. Popelinsky, M. Nepil (éditeurs), *FI MU Report serie, FI-MU-RS-2001-08*, p. 11–22, 2001.

[11] G. Bonfante, P. de Groote, « Stochastic Lambek Categorial Grammars », *in : Electronic Notes in Theoretical Computer Science, 53*, Elsevier Science Publishers, To appear.

[12] G. Bonfante, J.-Y. Marion, J.-Y. Moyen, « On termination methods with space bound certifications », *in : Andrei Ershov Fourth International Conference "Perspectives of System Informatics", Lecture Notes in Computer Science*, Springer, 2001. To appear.

[13] P. de Groote, « Strong Normalization of Classical Natural Deduction with Disjunction », *in : Fifth International Conference on Typed Lambda Calculi and Applications, TLCA'01*, S. Abramsky (éditeur), *Lecture Notes in Computer Science, 2044*, Springer Verlag, p. 182–196, 2001.

[14] P. de Groote, « Towards Abstract Categorial Grammars », *in : Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, p. 148–155, 2001.

[15] P. de Groote, « Type raising, continuations, and classical logic », *in : Proceedings of the Thirteenth Amsterdam Colloquium*, R. van Rooy, M. Stokhof (éditeurs), Institute for Logic Language and Computation, Universiteit van Amsterdam, p. 97–101, 2001.

[16] J.-Y. Marion, « Actual arithmetic and feasibility », *in : Computer Science Logic, Paris*, L. Fribourg (éditeur), *Lecture Notes in Computer Science, 2142*, Springer, p. 115–129, 2001.

[17] J.-Y. Moyen, « System Presentation : An Analyser of Rewriting Systems Complexity », *in : Electronic Notes in Theoretical Computer Science*, M. van den Brand, R. Verma (éditeurs), *59*, Elsevier Science Publishers, 2001.

[18] G. Perrier, « Intuitionistic Proof Nets as Models of Directed Acyclic Graph Descriptions », *in : 8th International Conference LPAR 2001, Havana, Cuba*, R. Nieuwenhuis, A. Voronkov (éditeurs), *Lecture Notes in Computer Science, 2250*, Springer, p. 233–248, 2001.

Rapports de recherche et publications internes

[19] G. Bonfante, F. Lamarche, « A model of a linear dependent calculus », *Rapport de recherche*, 2001.

[20] F. Lamarche, « On the Algebra of Structural Contexts », *Rapport de recherche*, 2001.