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

Responsable scientifique

Philippe de Groote [CR Inria]

Responsable permanent

François Lamarche [DR Inria]

Assistante de projet

Geneviève Pierrelée Grisvard

Personnel Universitaire

Adam Cichon [Professeur, Université Henri Poincaré]

Alain Lecomte [Professeur, Université de Grenoble 2]

Jean-Yves Marion [CR Inria, détaché de l'Université Nancy 2]

Guy Perrier [Maître de conférences, Université Nancy 2]

Guillaume Bonfante [ATER, Institut National Polytechnique de Lorraine]

Chercheurs doctorants

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

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

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

Jean-Yves Moyen [étudiant en quatrième année à l'ENS Lyon]

2 Présentation et objectifs généraux

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.

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, sémantique dénotationnelle, sémantique des jeux, complexité implicite.

3 Fondements scientifiques

Le thème de recherche de base du projet Calligramme est la logique linéaire : sa syntaxe, sa sémantique, ses rapports avec les formalismes constructifs traditionnels.

La logique linéaire, 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 (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. 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,

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		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*

n'est pas une primitive. Elle peut se définir, comme en logique classique, de la manière suivante : $A \multimap B \;=\; A^\perp B$.

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.

Les activités du projet Calligramme s'organisent autour de trois modules ou actions de recherche :

- calcul des séquents, réseaux de démonstration et d'interaction ;
- logique linéaire et grammaires formelles ;
- λ-calculs typés, sémantique, complexité.

Ces trois 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.

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

3.1 Calcul des séquents et réseaux de démonstration

Mots clés : calcul des séquents, réseau de démonstration.

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.

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.*

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 ;
- 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.

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

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 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.

3.2 Logique linéaire et grammaires formelles

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

[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.

[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.

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 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 :

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

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

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

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.

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.

3.3 Lambda calculs typés, sémantique, complexité

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

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

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*

[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.

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) et l'emploi de systèmes de types pour l'expression de classes de complexité. Nous étudions également les aspects sémantiques du lambda-calcul. 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.

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.

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, dûs 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.

[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.

[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.

Complexité implicite des calculs

La construction de logiciels sûrs est une nécessité. Il est tout aussi 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 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 de fonctions calculées par les programmes de L, *i.e.* , $\{[\![\mathtt{p}]\!]_{\mathtt{L}} : \mathtt{p}$ est un programme 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 génétiquement 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.

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 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

4.1 Panorama

Résumé : *Le projet Calligramme vise à appliquer ses compétences en logique linéaire au traitement automatique des langues, plus précisément à l'analyse syntaxique de textes (dans le but d'en produire une représentation sémantique) et à la génération automatique de textes à partir de données formelles. Le français est la langue privilégiée.*

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.

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 et réseaux de démonstration

Philippe de Groote a utilisé la théorie des réseaux de démonstration pour construire des algorithmes de démonstration automatique pour divers fragments implicatifs de la logique linéaire [19]. Il a montré comment la prouvabilité d'une formule linéaire implicative peut être réduite à un problème de couplage consistant à trouver, d'une part, une bijection entre deux ensemble de termes du premier ordre et, d'autre part, une substitution unifiant les termes mis en correspondance par cette bijection. Lorsque les termes du premier ordre prennent leur valeur dans un groupoide libre, un monoide libre, ou un monoide commutatif libre, le problème de couplage ainsi défini correspond, respectivement, à la prouvabilité au sein du calcul non-associatif de Lambek, du calcul associatif de Lambek, ou du fragment implicatif de la logique linéaire de Girard.

Philippe de Groote et François Lamarche ont défini une variante non-associative de la logique linéaire multiplicative [12]. Ils en ont donné le calcul des séquents, ont défini une notion originale de réseau de démonstration non associatif, et ont prouvé que le problème de décision associé est polynomial. Leur système peut être vu comme la version classique du calcul non-associatif de Lambek (NL).

5.2 Logique linéaire et grammaires formelles

Construction d'un modèle linguistique

Guy Perrier a continué de développer le modèle des *Grammaires d'interaction* [23], 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.

La spécificité de la logique linéaire intuitionniste permet de donner une représentation plus compacte des réseaux de démonstration qui ne prenne en compte qu'un ordre induit par ces réseaux sur les formules atomiques qui les composent.

A partir de là, on peut ramener une démonstration en logique linéaire intuitionniste à la recherche d'un modèle valide d'une description d'arbre polarisée. La description représente la structure de la formule à démontrer sous forme de relations de domination et de parenté entre formules atomiques polarisées qui la composent et le modèle de cette description est un arbre qui est obtenu en identifiant par paires duales les formules atomiques positives et négatives.

Les grammaires d'interaction sont une application de ce mécanisme à la syntaxe des langues [22] avec un certain relâchement quant à la façon dont les modèles sont obtenus à partir des descriptions : en plus de l'opération de branchement entre nœuds duaux, il est possible de superposer des branches d'arbres.

Maintenant, si l'on compare ce formalisme aux grammaires de Lambek, on constate qu'il a une souplesse plus grande liée notamment au fait que l'ordre linéaire entre constituants est dissocié de la relation de domination entre ceux-ci.

Pour prendre en compte toute la complexité de l'information linguistique qui contribue à la bonne formation des énoncés, les nœuds des descriptions sont associées à des matrices de traits et les polarités sont descendues du niveau des nœuds, c'est-à-dire des constituants, au niveau des traits qui les caractérisent. Ainsi, par exemple, le nœud représentant un nom propre sera muni d'un trait positif indiquant qu'il est capable de fournir la catégorie grammaticale *groupe nominal* et d'un trait négatif indiquant que ce groupe nominal attend de recevoir une fonction dans la phrase.

La composition syntaxique est alors réalisée sous forme de branchement et de superposition partielle d'arbres sous-spécifiés contrôlée par l'interaction "électrostatique" entre les traits.

Le développement, en cours, d'un prototype d'analyseur syntaxique du français sous Oz [Smo95] vise à tester la pertinence linguistique et computationnelle du formalisme. Les efforts actuels visent à étendre significativement la couverture linguistique de la grammaire utilisée par l'analyseur en vue de traiter de larges corpus. Une des questions clés est celle de l'organisation de cette grammaire et le choix d'une organisation en modules facilite à la fois la mise à jour et l'interrogation de la grammaire.

La vision de la composition syntaxique comme une interaction entre les mots de la langue a amené tout naturellement Guy Perrier à se pencher sur les liens qu'il pouvait y avoir avec

[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.

les *grammaires de dépendance* [Mel88]. Si cette ouverture en direction des grammaires de dépendance semble riche de perspectives, il est encore trop tôt pour en tirer des conclusions.

Par ailleurs, le travail mené par François Lamarche sur la nature des représentations linguistiques et leur formalisation mathématique a continué d'évoluer. Le besoin pour le traitement des langues de superposer plusieurs niveaux de représentation en une structure unifiée se révèle être un problème formel difficile. Une des raisons pour ceci est la multi-dimensionnalité des données qui doivent être intégrées, et le besoin d'une certaine notion de « continuité » ou de « connexité », qui est assez difficile à bien saisir, comme le montre les travaux de cognitivistes comme Gärdenfors. Une évolution importante du travail qui est apparue cette année est la rálisation de l'importance de la notion de rôle, au sens linguistique traditionnel du terme: rôle (ou fonction) syntaxique, rôle sémantique.....). Ceci a donné lieu à l'identification d'une classe de théories du premier ordre dont les modèles sont des structure arborescentes à branchement multidimensionnel (en fait, puisque le partage est permis, on va plus loin que la notion d'arbre). Le concept de rôle est essentiel pour définir ces structures, même s'il est employé dans un sens abstrait et très général. Cet état du travail a donné lieu à une communication écrite [20] au ESSLI2000 Workshop on Trees.

Génération automatique de textes

Peu de travaux se rapportent à la génération dans le cadre des grammaires de type logique et en particulier du calcul de Lambek, à l'exception notable de [MM96]. Néanmoins, la méthode proposée dans

cet article, basée sur l'unififcation de λ-termes, pose des problèmes de décidabilité. Il s'agissait donc de savoir si cette impossibilité découlait du cadre formel général dans lequel nous considérions la génération, ou bien si cette indécidabilité n'était pas dans une certaine mesure artificielle et propre à la méthode adoptée.

Pour montrer que le problème traité n'est pas intrinsèquement indécidable, nous avons opté pour une approche à partir des réseaux sémantiques définis par de Groote et Retoré [dGR96]. Celle-ci se base sur la recherche de preuve de la logique linéaire. Nous avons pu proposer une méthode originale de recherche de preuve qui d'une part montre la décidabilité de la réalisation syntaxique (une fois les items lexicaux choisis) et d'autre part caractérise les cas pour lesquels cette réalisation s'effectue efficacement (en temps polynomial).

Nous avons également poursuivi l'implémentation d'un logiciel capable d'utiliser ces résultats théoriques pour la génération. Les efforts ont porté sur des problèmes d'optimisation et de réorganisation mais doivent se concentrer désormais sur l'ajout des nouveaux éléments pour la génération. Néanmoins nous disposons maintenant d'un système capable de faire de l'analyse syntaxique en se basant sur les réseaux de preuve de la logique linéaire.

[Mel88] I. Mel'cuk, *Dependency Syntax: Theory and Practice*, Albany, N.Y.: The SUNY Press, 1988.

[MM96] J. Merenciano, G. Morrill, «Generation as deduction on labelled proof nets», *in: Logical Aspects of Computational Linguistics, LACL'96, Nancy, september 1996*, C. Retoré (éditeur), *Lecture Notes in Computer Science, 1328*, Springer Verlag, p. 310–328, 1996.

[dGR96] P. de Groote, C. Retoré, « On the Semantic Readings of Proof-Nets », *in: Formal Grammar*, G. M. Geert-Jan Kruijff, D. Oehrle (éditeurs), FoLLI, p. 57–70, Prague, août 1996.

5.3 Lambda calculs typés, sémantique, complexité

Lambda calculs typés

Guillaume Bonfante, Adam Cichon et François Lamarche s'intéressent à la complexité vue sous l'angle des termes ordinaux, autrement dit les ordinaux de Brouwer. Ces derniers, a l'instar des ordinaux, peuvent être utilisés pour mesurer la complexité d'une fonction. Vis-à-vis des ordinaux, les termes ordinaux offrent davantage de perspectives de par leur plus grande souplesse d'utilisation et de par des distinctions plus fines qu'ils permettent de faire entre deux processus.

Guillaume Bonfante et François Lamarche cherchent à définir un λ-calcul typé, une extension du système T de Gödel, dans lequel les termes ordinaux ont un type propre et dans lequel il y a un récurseur spécifique au principe d'induction des termes ordinaux. Simmons propose une telle approche, mais il escamote le problème de l'ordre, en particulier, il ne peut pas faire de comparaisons entre les termes ordinaux à l'intérieur de son système. Les travaux de Guillaume Bonfante et François Lamarche proposent une solution à ce problème.

Prendre en compte la notion d'ordre pose des problèmes sémantiques. En particulier, on trouvera dans [8] une nouvelle structure monoïdale sur les graphes, qui résout l'équation de la currification dans le cas où les types « sont des ordres[2] ». Cette structure sert pour définir une notion de structure algébrique, notion qui permet de définir des ordres classiques, comme l'ordre usuel sur les entiers naturels, une extension de l'ordre des ordinaux de Brouwer ou encore l'ordre de Kruskal.

Ces travaux sont le point de départ de la construction du calcul « bold » décrit dans [24]. Il s'agit d'un λ-calcul typé — avec des types dépendants — qui vérifie le paradigme « un type est un graphe ». Ils utilisent pour cela un calcul à deux niveaux, le premier correspond aux ensembles et le deuxième aux graphes. En particulier, ils étudient la notion de substitution dans ce cadre. Deux substitutions sont envisageables : une première, sémantique, qui s'appuie sur la substitution ensembliste et une deuxième, syntaxique, qui correspond à la substitution naïve, mais dont la définition ne l'est pas.

Philippe de Groote travaille à la définition d'une notion abstraite de grammaire catégorielle généralisant plusieurs formalismes grammaticaux existant. Ces grammaires catégorielles abstraites sont basées sur un lambda calcul typé linéaire. Dans ce cadre il a été amené a étudié le filtrage linéaire d'ordre supérieur. Il a démontré que ce problème était décidable et NP-complet [18].

Logique classique et opérateurs de contrôle

Catherine Pilière termine actuellement la rédaction d'un travail de thèse portant sur une approche statique du traitement d'exceptions en programmation fonctionnelle. Ses recherches concernent l'extension via un opérateur de point fixe, d'un formalisme introduit par Philippe de Groote [dG95]. Il s'agit au départ d'un λ-calcul simplement typé doté d'un mécanisme de

2. En fait des graphes.

[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*

contrôle assurant qu'aucune exception ne peut échapper à son champ de déclaration au cours de l'exécution d'un programme correctement typé. L'ajout d'un opérateur de point fixe au calcul initial permet d'étudier le paradigme dans un cadre de programmation réaliste autorisant la récursion générale. Le calcul enrichi de l'opérateur de récursion 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 [25].

Sémantique dénotationnelle et sémantique des jeux

La logique linéaire non commutative a été introduite dans la thèse de Paul Ruet en 1997; il est donc né à ce moment un programme de recherche qui vise en premier à étendre à ce système tous les outils que possède la logique linéaire: calcul des séquents, théorie des réseaux de démonstration, sémantique des phases... paradoxalement, l'aspect qui a résisté le plus longtemps est celui de la sémantique dénotationelle. Un travail de collaboration entre Rick Blute, François Lamarche et Paul Ruet s'attaque à ce problème depuis l'été 1998. Ce travail a donné lieu à deux solutions distinctes, mais néanmoins proches du point de vue des techniques employées, et qui sont présentées dans un seul exposé commun. Le point commun est la construction de structures algébriques munies de deux opérations associatives, l'une commutative et l'autre non commutative. La différence principale est le choix d'une étape intermédiaire. Dans un cas, on passe par un modèle du calcul intuitionniste de de Groote, qu'on transforme en modèle classique au moyen d'une version de la « construction de Chu » adaptée à ce nouveau contexte; dans l'autre cas, on utilise comme étape intermédiaire un modéle classique, mais où les tenseurs et les pars sont identifiés ; le processus de « séparation » des connecteurs se fait au moyen d'un type particulier d'algèbre de Hopf. Le travail [10] a été soumis à *Theory and Applications of catégories.*

François Lamarche a mis au point un type nouveau de sémantique pour la logique linéaire [26], qu'il a nommé « jeux de déplacement ». L'idée de base est l'action mutuelle de deux ensembles l'un sur l'autre. On peut voir ces ensembles comme des stratégies pour deux joueurs, ou des tentatives de preuve et de réfutation d'une formule. Une stratégie s réussit lorsque toutes les contre-stratégies «ne réussissent pas à déplacer s», autrement dit que s est un point fixe pour leurs actions. En fait on peut remplacer les ensembles par n'importe quel modèle catégorique du lambda-calcul. L'interprétation des connecteurs dans ce modèle a plusieurs points en commun avec la "construction de Chu" de Barr et le modèle Dialectica de de Paiva. Le modèle de base obtenu identifie l'unité du tenseur et celle du par («règle Mix»), mais au moyen des techniques d'actions de monoïdes mises au point par Barr pour la construction de Chu, on peut remédier à ceci. Il reste à explorer l'intérêt de ce modèle du point de vue des théorèmes de complétude.

in Computer Science, 902, Springer Verlag, p. 201–215, 1995.

Complexité implicite des calculs

Le mémoire d'habilitation de Jean-Yves Marion [9] fait un tour d'horizon, de la théorie aux applications, de la complexité implicite des calculs. Une première direction consiste à analyser la complexité algorithmique, d'une spécification algébrique, *i.e.* de déterminer la quantité de ressources nécessaires pour réaliser une spécification. Pour cela, les ordres de terminaison utilisés en réécriture ont été choisis comme outils d'analyse. Une seconde direction concerne les logiques constructives (théorie des types de Martin-Löf [ML84], calcul des constructions [CH88]). Ces théories permettent de synthétiser des programmes sûrs à partir d'une démonstration de leurs corrections. Des systèmes tels que Alf, Coq et Nuprl fonctionnent sur ce principe. Dans ce cadre, l'apport de Jean-Yves Marion est la possibilité de certifier la complexité du programme synthétisé [27].

Guillaume Bonfante, Adam Cichon, Jean-Yves Marion et Hélène Touzet ont étudié la complexité des programmes dont la terminaison est établie par une interprétation polynomiale [11]. Ils ont montré comment caractériser, en fonction du type d'interprétation polynomiale associée aux constructeurs, des classes de fonctions calculables en temps déterministe : polynomial, exponentiel, et doublement exponentiel. Des classes fonctions calculables en temps non-détermiste (polynomial, exponentiel, et doublement exponentiel) ont également été caractérisées en considérant des systèmes de réécriture non-confluents. Enfin, ils ont montré que la classe des fonctions programmable sous la forme d'un système qui termine par une interprétation exponentielle est exactement la classe des fonctions élémentaires.

Un prolongement de ces travaux est exposé dans la thèse de Guillaume Bonfante [8]. Il y est démontré que les quasi-interprétations (c'est-à-dire les interprétations décroissantes et non strictement décroissantes) donnent lieu à une hiérarchie en espace : espace linéaire, polynomial, exponentiel et doublement exponentiel. Guillaume Bonfante s'est également intéressé à l'ordre KBO [16]. La définition de contraintes syntaxiques sur cet ordre lui a permis de caractériser trois classes de complexité : LINSPACE, Space(n^2) et ESPACE.

Dans [15], Jean-Yves Marion a défini un nouvel ordre de terminaison appelé *light multiset path ordering* (LMPO), qui est une restriction de l'ordre de terminaison *multiset path ordering*. Un programme qui termine par LMPO représente une fonction calculable en temps polynomial. A la suite de ce travail Jean-Yves Marion et Jean-Yves Moyen ont montré que les programmes qui terminent par l'ordre de terminaison *Multiset path ordering* et qui admettent une quasi-interprétation polynomiale définissent exactement la classe des fonctions calculables en temps polynomial [21]. L'intérêt de ces deux résultats tient aux faits qu'ils capturent des algorithmes pratiques, permettent d'optimiser les programmes en utilisant la programmation dynamique comme méthode de compilation, et certifient la complexité du programme. Un point qui est parfois difficile à comprendre, au premier abord, mérite d'être souligné : ces résultats portent, en partie, sur l'analyse de la complexité implicite d'un programme et non de sa complexité explicite. De ce fait, l'analyse permet de tranformer le programme source en un programme plus efficace.

Adam Cichon et Jean-Yves Marion [17] ont construit une restriction de l'ordre de termi-

[ML84] P. Martin-Löf, *Intuitionistic Type Theory*, Bibliopolis, 1984.

[CH88] T. Coquand, G. Huet, «The Calculus of Constructions», *Information and Computation 76*, 1988, p. 95–120.

naison *Lexicographic path ordering*. Les programmes qui terminent par cet ordre définissent exactement les fonctions calculables en espace polynomial.

Jean-Yves Marion dans [27] a proposé une restriction de l'arithmétique de Peano, qui caractérise les fonctions calculables en temps polynomial. Cette restriction porte sur la quantification universelle qui est limitée à une catégorie de termes (dits canoniques suivant dans la terminologie de Martin-Löf).

Enfin, l'article [14] de Daniel Leivant et Jean-Yves Marion caractérise la classe NC_1. Cette classe est celle des fonctions calculables par des familles de circuits booléens uniformes de profondeur logarithmique et de taille polynomiale, dans la taille de l'entrée. Elle est équivalente à la classe ALOGTIME des fonctions calculables sur machines de Turing alternantes en temps logarithmique.

Apprentissage et Complexité algorithmique de l'information

Serge Grigorieff et Jean-Yves Marion [13] ont étudié la complexité de Kolmogorov dans le cas de mots produits par un calcul non-déterministe. C'est la première fois que cette problématique est explicitement posée. Pour cela, ils considèrent cinq classes de modes de description non-déterministe :

1. des modes bornés (*bounded modes*) pour lesquels un programme ne génère qu'un nombre fini de mots.

2. des modes distribués (*distributed modes*) pour lesquels le nombre de sorties d'un programme dépend de la taille de ces sorties.

3. des modes étalés (*spread modes*) pour lesquels le nombre de sortie d'un programme dépend à la fois du programme et de la la taille des sorties.

4. des modes discriminants (*discriminating modes*) dans lesquels un mot à une description minimale unique.

5. des modes pour lesquels l'ensemble des descriptions de longueurs minimales forme un ensemble préfixe.

Jérome Besombes et Jean-Yves Marion étudient des extensions du modèle d'apprentissage à la limite de Gold. Leur approche s'appuie sur la théorie algorithmique de l'information. Le modèle construit correspond à celui d'un professeur en face d'une classe d'étudiants pouvant communiquer. Dès lors, la classe a appris si un seul élève a appris.

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. 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 régionales

Jean-Yves Marion anime l'action : *Propriétés non fonctionnelles des composants : fiabilité et complexité* du Contrat Plan Etat Region (CPER) : Qualité et sureté du logiciel.

Calligramme est partie prenante de la thématique *Ingénierie des Langues, du Document, de l'Information Scientifique et Culturelle* du contrat de plan Etat Région.

7.3 Actions européennes

Réseau TMR : « Logique linéaire et Informatique »

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).

Action intégrée ALLIANCE : « Complexité implicite des calculs et théorie des algorithmes »

Participants : Guillaume Bonfante, Adam Cichon, Jean-Yves Marion, Jean-Yves Moyen, Iain Stewart et Florent Madeleine de l'université de Leicester, Angleterre.

Visite d'Adam Cichon et de Jean-Yves Marion à Leicester, Angleterre en août 2000.

Action intégrée Van Gogh : « Logiques des ressources, Réseaux de Démonstration et Analyse de la Langue »

La tranche finale de l'action intégrée Van Gogh « Logiques des ressources, Réseaux de Démonstration et Analyse de la Langue » s'est conclue en Juin 2000. Ce projet, coordonné à Nancy par François Lamarche et à Utrecht par Michael Moortgat, visait à l'étude des logiques substructurales du point de vue des réseaux de démonstrations. En 2000, il y a eu des séjours d'une semaine de Quentijn Puite et Richard Moot à Nancy, et de Francois Lamarche à Utrecht.

7.4 Visites, et invitations de chercheurs

Bruno Guillaume a donné un séminaire dans le cadre du groupe de travail de calligramme, le 11 Avril 2000.

Jean-Yves Marion et Adam Cichon ont reçu Iain Stewart et Florent Madeleine, du 26 Juin au 1er Juillet 2000, dans le cadre du projet Allaince.

Guy Perrier a reçu Denys Duchier de l'université de Sarrebrück les 6 et 7 décembre 2000. Ce dernier a effectué un exposé sur le thème *dépendance, précédence et contraintes* au séminaire du pôle *Ingénierie des Langues, du Document, de l'Information Scientifique et Culturelle* du LORIA.

8 Diffusion de résultats

8.1 Animation de la Communauté scientifique

Worskshop et groupe de travail

Jean-Yves Marion a été organisateur du workshop international *Implicit computational complexity*, ICC'00, affilié à la conférence LICS, du 29 au 30 Juin. A la suite de ce workshop, un numéro spécial du journal *Theoretical Computer Science* sera édité.

Jean-Yves Marion a été membre du comité de programme du workshop international *Rule-Based Programming*, affilié à la conférence PLI2000, 19 Septembre 2000, Montreal, Canada.

Jean-Yves Marion a été membre du comité de programme des journées sur la théorie algorithmique de l'information, TAI'00, 09 Juin 2000, Lille.

Catherine Pilière a présidé le comité de programme de la conférence *ESLLI 2000 Student Session*.

8.2 Enseignement

DEA

Adam Cichon a enseigné le module de Techniques Avancées « Logique 2 » sur le théorème de Gödel pour le DEA Informatique 1999-2000.

Adam Cichon a enseigné l'option « logique et démonstration automatique » pour le DEA Informatique 2000-2001.

Philippe de Groote a enseigné l'option « calculs pour la modélisation et la preuve » pour le DEA Informatique 2000-2001.

Jean-Yves Marion a enseigné l'option « modèles de calcul et complexité » pour le DEA Informatique 2000-2001.

Guy Perrier a enseigné l'option « analyse syntaxique des langues» pour le DEA Informatique 2000-2001.

DESS

Adam Cichon a enseigné le module « Logique et Programmation » pour le DESS Informatique Double Compétence.

Encadrement Doctoral

La thèse de Guillaume Bonfante [8], encadrée par Adam Cichon, a été soutenue en Octobre 2000.

Philippe de Groote encadre la thèse de Catherine Pillière, « Une approche contrôlée du traitement d'exceptions en programmation fonctionnelle », depuis Septembre 1997.

Jean-Yves Marion encadre la thèse de Jean-Yves Moyen,« Analyse de la complexité des programmes et transformation de programmes », depuis Septembre 2000.

Jean-Yves Marion encadre la thèse de J. Besombes (boursier MESR), « Apprentissage des grammaires catégorielles et complexité de Kolmogorov », depuis Septembre 1999.

8.3 Jurys de thèse

François Lamarche a été rapporteur interne de la thèse de Horatiu Cirstea. Thèse soutenue à l'Université Henri Poincaré 27 Octobre 2000.

François Lamarche sera rapporteur de la thèse de Quentijn Puite. Thèse soutenue à l'Universitéd'Utrechtle 26 Janvier 2001.

Jean-Yves Marion a été rapporteur de la thèse de Jean-Christophe Dubacq (sous la responsabilité de Adam Cichon), directeur de thèse Bruno Durand. Thèse soutenue à l'École normale supérieure de Lyon le 29 Septembre 2000.

Jean-Yves Marion a été membre du jury de thèse de G. Bonfante, directeur de thèse Adam Cichon. Thèse soutenue l'Université Henri Poincaré le 27 Octobre 2000.

8.4 Participation à des colloques, séminaires, invitations

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

Philippe de Groote a effectué un séjour de recherche de sept mois à l'Université polytechnique de Catalogne, à Barcelone. Il y a travaillé en collaboration avec Glyn Morrill (Mars à Septembre 2000)

Sylvain Pogodalla a participé à la conférence NAACL 2000, à Seattle, du 26 Avril au 4 mai 2000. Il y a présenté une communication [**?**].

Jérome Besombes et Guy Perrier ont participé aux rencontres « grammaire et logique », à Rennes, du 2 au 5 Mai 2000.

Guy Perrier a participé à la conférence TAG+5, à Paris, du 25 au 27 Mai 2000. Il y a présenté une communication [22].

Jérome Besombes et Jean-Yves Marion ont participé aux journées T.A.I., à Villeneuve d'Ascq, du 7 au 9 Juin 2000.

Jérome Besombes, Guillaume Boinfante, Jean-Yves Marion et Jean-Yves Moyen ont participé à la conférence LICS 2000, à Santa Barbabra, du 24 Juin au 2 Juillet 2000. Guillaume Boinfante a présenté une communication [16] à ICC, *workshop* satellite de LICS.

François Lamarche s'est rendu à Utrecht, du 25 au 28 Juin 2000, dans le cadre du projet Van Gogh.

Philippe de Groote a participé à la conférence RTA 2000, à Norwich, du 8 au 13 Juillet 2000. Il y a présenté une communication [18].

Catherine Pilière a participé au *workshop "foundations and Computations"* , à Edimbourg, du 15 au 19 2000. Elle y a présenté une communication [25].

Philippe de Groote, Sylvain Pogodalla et Guy Perrier ont participé à la conférence COLING 2000, à Sarrebruck, du 31 Juillet au 4 Aout 2000. Sylvain Pogodalla y a présenté une communication [?].

Adam Cichon et Jean-Yves Marion se sont rendus à Leicester du 2 au 8 Aout 2000, dans le cadre du projet Alliance.

Jean-Yves Moyen, Catherine Pilière et Sylvain Pogodalla ont participé à l'école d'été ESLII 2000, à Birmingham, du 05 au 20 Aout 2000.

Adam Cichon a participé à la conférence PLI 2000, à Montreal, du 16 au 20 Septembre 2000. Il y a présenté une communication [17] au *workshop RULE 2000*, associé à cette conférence.

Philippe de Groote et Jean-Yves Moyen ont participé à la conférence LPAR 2000, à La Réunion, du 5 au 12 Novembre 2000. Ils y ont présenté chacun une communication [19, 21].

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.

Thèses et habilitations à diriger des recherches

[8] G. Bonfante, *Constructions d'ordres, analyse de la complexite*, Thèse d'université, LORIA, octobre 2000.

[9] J.-Y. Marion, *Complexité implicite des calculs, de la théorie à la pratique*, Habilitation à diriger des recherches, décembre 2000.

Articles et chapitres de livre

[10] R. F. Blute, F. c. Lamarche, P. Ruet, « Entropic Hopf Algebras and Models of Non Commutative Logic », *Theory and Applications of Categories*, 2001.

[11] G. Bonfante, A. Cichon, J.-Y. Marion, H. Touzet, « Algorithms with Polynomial Interpretation Termination Proof », *Journal of functional programming 11*, septembre 2000, p. 1–21.

[12] P. de Groote, F. c. Lamarche, «Classical Non Associative Lambek Calculus», *Studia Logica*, 2000.

[13] S. Grigorieff, J.-Y. Marion, «Kolmogorov complexity and non-determinism», *Theoretical Computer Science*, octobre 2000.

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

[15] J.-Y. Marion, «Analysing the implicit complexity of programs», *Information and Computation*, novembre 2000.

Communications à des congrès, colloques, etc.

[16] G. Bonfante, «Two complexity classes within KBO», *in : , Santa Barbara, US*, mai 2000.

[17] E. Cichon, J.-Y. Marion, « The Light Lexicographic path Ordering», *in : , Montréal, Canada*, septembre 2000.

[18] P. de Groote, « Linear higher-order matching is NP-complete», *in : Rewriting Techniques and Applications, Norwich*, L. Bachmair (éditeur), *Lecture Notes in Computer Science*, *1833*, Springer, p. 127–140, juillet 2000.

[19] P. de Groote, «Proof-search in implicative linear logic as a matching problem», *in : Logic for Programming and Automated Reasoning, Reunion Island*, A. V. Michel Parigot (éditeur), *Lecture Notes in Artificial Intelligence*, *1955*, Springer, p. 257–274, novembre 2000.

[20] F. Lamarche, « Beyond Trees», *in : ESSLI2000 Workshop on Trees, Birmingham, UK*, V. Goranko (éditeur), V. Goranko, 2000.

[21] J.-Y. Marion, J.-Y. Moyen, « Efficient first order functional program interpreter with time bound certifications», *in : LPAR*, *Lecture Notes in Computer Science*, *1955*, Springer, p. 25–42, 2000.

[22] G. Perrier, « From Intuitionistic Proof Nets to Interaction Grammars», *in : TAG + 5, Paris*, Université Paris 7, mai 2000.

[23] G. Perrier, « Interaction Grammars», *in : CoLing 2000, Sarrebrück*, International Committee on Computational Linguisitcs, août 2000.

Rapports de recherche et publications internes

[24] G. Bonfante, F. Lamarche, « A monotone lambda-calculus : the example of ordinal terms», *Rapport de recherche*, octobre 2000.

[25] P. de Groote, C. Piliere, « On the semantics of static exception handling », *Rapport de recherche*, octobre 2000.

[26] F. Lamarche, «Displacement Games as a model of linear logic», *Rapport de recherche*, novembre 2000.

[27] J.-Y. Marion, « A proof theoretic approach to feasible computation », *Rapport de recherche*, février 2000.