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 [DR INRIA]

Responsable permanent
François Lamarche [DR INRIA]

Assistante de projet
Laurence Benini

Personnel INRIA
Bruno Guillaume [CR]
Sylvain Pogodalla [CR à partir du 01.11.02]

Personnel Universitaire
Adam Cichon [Professeur, Université Henri Poincaré]
Jean-Yves Marion [Maître de conférences, Université Nancy 2 jusqu'au 28.02.02, Professeur, Ecole des Mines de Nancy à partir du 01.03.02]
Guy Perrier [Maître de conférences, Université Nancy 2, détaché au CNRS]
Guillaume Bonfante [Maître de conférences, Ecole des Mines de Nancy]

Chercheurs doctorants
Jérôme Besombes [allocataire MESR, Université Henri Poincaré, Moniteur à l'Université de Metz]
Paulin Jacobé de Naurois [allocataire normalien, thèse coencadrée par les projets Protheo et Calligramme]
Jean-Yves Moyen [allocataire normalien]
Sylvain Salvati [allocataire MESR]

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

3.1. Introduction

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 [26], résulte d'une analyse fine du rôle joué par les règles structurelles dans le calcul des séquents de Gentzen[25]. 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 1),

Table 1. Les connecteurs de la logique linéaire

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

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 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 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[21] 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.

3.2. 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, Bruno Guillaume, François Lamarche, Guy Perrier.

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.

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 naturelle 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 [27].

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.

3.3. 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.

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 :

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

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 [18], 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 [29][30]. 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.

3.4. 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.

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 LPL et de démontrer que la classe des fonctions calculées par les programmes de LPL, est exactement la classe Ptime. Ce type de résultat ne répond pas aux questions de l'expert en programmation, car le langage de programmation LPL 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'application

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 théorie 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

5.1. 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 les grammaires d'interaction. Une première version de ce logiciel, utilisant la programmation par contraintes, avait été développée par Guy Perrier.
  L'analyseur a été ré-écrit en CAML par Guillaume Bonfante, Bruno Guillaume et Guy Perrier. Il intègre les derniers résultats sur l'analyse syntaxique avec les grammaires d'interaction (voir section 6.2.1) :
  - Une première passe permet de filtrer de façon drastique les choix lexicaux possibles selon deux critères : un qui utilise la neutralité globale des polarités intervenant dans l'analyse et un autre local à la Brill (on élimine des analyses contenant des séquences interdites en français : par exemple un déterminant ne peut pas être suivi d'un verbe).
  - Dans la passe d'analyse proprement dite, des heuristiques, qui s'appuient sur des considérations psycholinguistiques, permettent d'élaguer efficacement des branches de l'arbre de recherche.

  L'analyseur fonctionne actuellement avec une grammaire réduite gérée manuellement. L'ambition est de l'utiliser prochainement avec une grammaire à large couverture générée automatiquement à partir d'une méta-grammaire. Une méta-grammaire [15] est un ensemble de modules représentant les constructions grammaticales de la langue sous forme de descriptions d'arbres. Ces modules prennent place au sein d'un système d'héritage multiple permettant une factorisation importante de l'information.

Une seconde ambition est d'intégrer la sémantique à la grammaire pour que l'analyseur retourne une représentation sémantique des phrases analysées. Dans un premier temps, cela pourrait se faire simplement en utilisant une sémantique plate [19].
Enfin, le passage à l'échelle au niveau de la grammaire nécessitera d'améliorer les performances de l'analyseur. Pour élargir les applications possibles, il faudra aussi le rendre robuste, c'est-à-dire faire en sorte que toute phrase même incorrecte donne lieu à une analyse partielle, ce que permet tout à fait le formalisme des grammaires d'interaction.

- **ICAR** est un logiciel qui permet d'analyser la complexité implicite de certains systèmes de réécriture. Il est décrit plus amplement dans [31]. **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.
- **ALTEA** est une logiciel d'apprentissage de formalismes linguistiques écrit en langage Java. Le moteur d'inférence est l'algorithme d'apprentissage de langages d'arbres réversibles à partir d'exemples positifs et s'applique automatiquement au cas des langages d'arbres, des langages algébriques, des langages de dépendances et des langages générés par des grammaires catégorielles. Une version de ce logiciel sera prochainement disponible en ligne sous forme d'applet.
- **LAMBDAMU** est un interprète interactif d'un $\lambda\mu$-calcul simplement typé avec appel par valeur, sur lequel est basée une nouvelle forme de sémantique à la Montague.

6. Résultats nouveaux

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

Le projet d'unification des logiques sub-structurelles lancé par François Lamarche l'année précédente a progressé en 2002 dans plusieurs directions. La plus visible de celles-ci est une caractérisation purement algébrique, au moyen de la théorie des catégories, de la notion de « logique sub-structurelle ». Rappelons que l'an 2001 a vu la naissance de deux concepts, soient celui de structade (théorie des contextes pour une logique sub-structurelle) et celui de « framework » (choix de connecteurs pour une structade, qui engendre mécaniquement un calcul des séquents, doté de l'élimination des coupures). Ces concepts unifient de façon systématique toutes les logiques sub-structurelles purement multiplicatives qui peuvent se formuler dans le calcul des séquents. Pour certaines d'entre elles une formulation au moyen de la théorie des catégories existe déjà depuis longtemps ; par exemple la logique linéaire multiplicative intuitionniste correspond aux catégories symétriques monoïdales fermées, la logique linéaire classique aux catégories *-autonomes, etc.. Dans [12], toutes ces définitions se retrouvent comme des cas particuliers d'une notion générale de « théorie fibrationnelle », qui est une version entièrement algébrique du systéme déductif engendré par un « framework ».

On peut voir la théorie des théories fibrationnelles comme une forme d'algèbre universelle dans les catégories. Deux aspects de cette approche sont novateurs. D'abord le fait que les problèmes de variance disparaissent tout simplement. Il s'agit du premier traitement satisfaisant des variances mixtes dans l'algèbre universelle des catégories, problème qui est resté sans solution pendant plus de trente ans. L'autre aspect intéressant est que les modèles d'une théorie fibrationnelle ne sont pas définis au moyen d'opérations strictes, mais plutôt avec des propriétés universelles, ce qui est toujours la façon naturelle de faire les choses dans ce contexte, pour des raisons techniques autant que conceptuelles. En fait comme le nom l'indique, une théorie fibrationnelle est l'extension de la notion de fibration de Grothendieck à des opérateurs d'arité arbitraire. Cette idée avait déjà été réalisée par Claudio Hermida dans le cadre beaucoup plus restreint des catégories monoïdales.

6.2. Grammaires catégorielles

6.2.1. Grammaires d'interaction

Guy Perrier a continué à développer le modèle des *grammaires d'interaction* dans l'objectif de montrer qu'il pouvait constituer une alternative crédible aux formalismes linguistiques existants [13].
L'originalité de ce modèle réside dans la notion de polarité qui fonde le principe de composition syntaxique : les syntagmes sont porteurs de traits positifs, négatifs ou neutres qui décrivent leurs propriétés morpho-syntaxiques et leur composition est guidée par la neutralisation des traits de même type mais de polarités opposées. La souplesse du modèle est due au fait qu'il utilise des descriptions d'arbres pour représenter des structures syntaxiques sous-spécifiées.
Un des avantages des grammaires d'interaction, du point de vue de leur pouvoir expressif, est qu'elles permettent d'effectuer de la superposition d'arbres et pas seulement de la substitution d'arbres comme la plupart des formalismes manipulant des arbres.
Guillaume Bonfante, Bruno Guillaume et Guy Perrier ont montré qu'il était possible de faire de l'analyse syntaxique efficace avec les grammaires d'interaction en s'appuyant sur la notion de polarité.
Celle-ci permet d'effectuer un filtrage redoutable des choix lexicaux préliminaires à l'analyse syntaxique proprement dite. Ce filtrage repose sur la neutralité globale des polarités présente dans les ressources lexicales utilisées pour analyser une phrase.
Ensuite, le processus d'analyse proprement dit utilise comme opération élémentaire la fusion de deux nœuds syntaxiques d'une description porteurs de deux traits de même type mais de polarités opposées. Cette opération est source d'explosion combinatoire du processus et pour contenir cette explosion, on utilise des heuristiques fondées sur des considérations psycholinguistiques qui visent à déclencher les neutralisations le plus tôt possible.
Un des enjeux est de montrer que les stratégies d'analyse qui ont été testées sur une grammaire restreinte du français avec le logiciel AGIR (cf. section 5.1) résistent au passage à une grammaire à large couverture.
La conception et la réalisation d'une telle grammaire nécessite tout un travail de recherche sur la meilleure de façon de factoriser l'information linguistique qu'elle contient pour la rendre manipulable à la fois par le linguiste et l'informaticien. C'est l'objet du travail autour de la notion de méta-grammaire qu'a commencé à mener Guy Perrier [15]. Un groupe de travail commun avec l'équipe Langue et Dialogue du LORIA a également réflechi sur le problème des méta-grammaires (4 séances en 2002).

6.2.2. Grammaires catégorielles abstraites

L'étude et le développement des grammaires catégorielles abstraites s'est poursuivi. Nos efforts ont porté sur le pouvoir d'expression des ACG et sur la complexité de l'analyse. Philippe de Groote a montré comment les grammaire d'arbres adjoints peuvent être représentées sous la forme d'ACG [11]. Bruno Guillaume et Guillaume Bonfante ont montré que les langages de mots générés par une ACG ne sont pas semi-linéaires. Sylvain Salvati a montré que même dans les cas les plus simples, le filtrage linéaire d'ordre supérieur restait NP-complet. Ces deux derniers résultats sont en cours de rédaction.

Le travail entrepris dans [22] a été poursuivi par Philippe de Groote et Chris Barker, de l'Université de San Diego, lors du séjour de ce dernier à Nancy, en juillet 2002. Ils ont développée un nouveau genre de sémantique de Montague basée sur le $\lambda\mu$-calcul. Un article est en cours de soumission.

6.2.3. Apprentissage grammatical

Des linguistes comme Chomsky et Pinker [32]ont montré que l'acquisition des langages naturels est basée sur l'analyse des structures de phrases correctes. Plus encore, il apparaît que la structuration des exemples est un élément prépondérant de ce processus ; en effet, appréhender les phrases entendues non pas comme une simple suite de mots, mais comme un arbre étiqueté par ces mots, permet la prise en compte d'informations syntaxiques et/ou sémantiques primordiales. L'acquisition du langage se révèle alors plus efficace. Dans le but de la compréhension et de la formalisation de ces phénomènes, Jean-Yves Marion et Jérôme Besombes se sont intéressés à l'apprentissage des grammaires régulières d'arbres[20]. Ces grammaires, basées sur les grammaires algébriques, sont représentées par un ensemble de règles dont la partie gauche est une variable et

la partie droite un arbre dont les nœuds sont étiquetés par des variables ou des symboles terminaux (mots du vocabulaire). Les langages générés par ces grammaires sont les ensembles d'arbres obtenus à partir d'une variable de départ, par remplacements successifs d'une variable d'une partie gauche d'une règle par la partie droite de cette règle. Les arbres obtenus ne sont plus étiquetés que par des mots. Apprendre de tels langages revient à reconstruire une grammaire à partir d'une séquence finie d'éléments d'un langage inconnu. Jean-Yves Marion et Jérôme Besombes ont montré qu'une restriction sur les règles (grammaires réversibles) est suffisante à l'apprentissage dans le modèle d'identification à la limite de Gold et ils ont donné un algorithme d'apprentissage efficace. Ils ont ensuite montré que leurs résultats sur l'apprentissage des langages d'arbres offrent des preuves inédites de plusieurs résultats importants du domaine de l'apprentissage des langages de mots (travaux de Sakakibara [34] sur les grammaires algébriques, de Kanazawa [28] sur les grammaires catégorielles, ...). Pour le cas le plus général des grammaires régulières d'arbres (la classe des grammaires sans la restriction de reversibilité), ils ont montré que l'apprentissage reste possible avec l'aide d'un oracle répondant à des questions d'appartenance d'un arbre au langage cherché. Cet oracle (le professeur) commence par fournir à l'algorithme d'apprentissage (l'étudiant), un ensemble d'éléments du langage puis ce dernier construit de nouveaux arbres qu'il soumet au professeur sous forme de questions d'appartenance. En fonction des réponses, l'étudiant construit une grammaire et le processus se termine dès que l'étudiant est convaincu que la gammaire construite est correcte. Jean-Yves Marion et Jérôme Besombes ont developpé un algorithme original résolvant ce problème et là encore, ce travail offre des preuves inédites de plusieurs résultats importants relatif aux langages de mots [16][17][33].

Les grammaires régulières d'arbres représentent un cadre théorique particulièrement adapté au cas des langages structurés. La prise en compte d'un ordre sur l'ensemble des nœuds des éléments des langages d'arbres permet d'adapter les algorithmes d'apprentissage au cas des grammaires de dépendances projectives. Cette classe de grammaires introduite par Hays et Gaifman[24], puis Dikovski et Modina [23] est largement utilisée dans le domaine linguistique. Les algorithmes développés dans le projet s'adaptent tout aussi naturellement aux domaines suivants :

- traitement de documents XML,
- recherches dans des bases de données,
- calcul de Lambek pour l'inférence linguistique.

6.3. Complexité implicite des calculs

6.3.1. Complexité des programmes fonctionnelles du 1erordre

Calculer la complexité des programmes est une question délicate. Notre point de vue est que l'on peut réutiliser un objet plus simple, les preuves de terminaison pour évaluer cette complexité. L'idée vient du fait que les méthodes de preuve de terminaison ont été largement développées, qu'elles couvrent correctement de nombreux algorithmes, et qu'elles offrent par la même un formidable outil pour analyser la complexité d'un programme fonctionnel du 1erordre.

Par l'analyse des preuves de terminaison, nous sommes capables de déterminer la complexité (e.g temps ou espace polynomial) d'un programme, à l'aide d'une analyse prédicative.

Ainsi 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 puissantes pour capturer des classes d'algorithme plus importantes. Guillaume Bonfante et Jean-Yves Marion travaillent actuellement sur des modèles de calculs qui évaluent plus finement les questions de complexité comme les Abstract State Machines de Y. Gurevich.

Dans le même ordre d'idée, Adam Cichon, en collaboration avec E. Tahhan-Bittar, poursuit sa recherche sur la complexité des systèmes de réécriture strictement orthogonaux. 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. Il travaille actuellement à une extension de cette théorie logique qui permettrait de raisonner sur les algorithmes.

8. Actions régionales, nationales et internationales

8.1. Actions régionales

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

8.2. Actions nationales

Calligramme participe à l'Action de Recherche Coopérative RLT (Ressources Linguistiques pour les TAG) dont les autres participants sont le projet Atoll de l'INRIA Rocquencourt, le projet Langue et Dialogue du LORIA et l'équipe TALaNa de Paris 7. Cette action a notamment permis de faire émerger la notion de méta-grammaire comme thème important à approfondir pour avancer sur l'organisation des grammaires à large couverture.

8.3. Visites et invitations de chercheurs

- Chris Barker, du département de linguistique de l'Université de Californie à San Diego, a séjourné dans le projet (1-31 juillet, 2002).
- Denys Duchier est venu dans le projet pour deux journées de travail en avril.
- François Lamarche a séjourné 4 semaines à Ottawa et Montréal, où il a travaillé avec Joachim Lambek, Michael Barr et Rick Blute.
- Jean-Yves Moyen a effectué une visite de 3 mois à l'université de Copenhague pour travailler avec Neil Jones dans le cadre de l'opération QSL ACT.

9. Diffusion des résultats

9.1. Animation de la Communauté scientifique

- Adam Cichon est suppléant pour la commission de spécialistes de l'UHP (27^{e} section).
- Philippe de Groote est membre titulaire élu de la commission d'évaluation de l'INRIA, vice président 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 Wollic'02 (30 juillet - 2 aout, 2002, Rio de Janeiro, Brésil). Il est membre du comité éditorial de la collection *Research Papers in Formal Linguistics and Logic* (éditée par l'Université de Bologne).

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

- François Lamarche est membre du bureau du Département de Formation Doctorale de l'école IAE+M, et président du comité des bourses de l'INRIA-Lorraine. Il était membre du comité d'organisation du symposium international PILM2002 (Philosophical insights into Logic and Mathematics) donné à l'Université Nancy 2, 30 septembre-4 octobre.
- Jean-Yves Marion et Jérôme Besombes ont organisé une rencontre sur l'apprentissage des grammaires catégorielles dans le cadre de l'Action de Recherche Coopérative INRIA GRACQ les 6 et 7 novembre à Nancy
- Guy Perrier est membre élu du conseil de laboratoire du LORIA. Il est membre du conseil des opérations du thème ILD&ISTC du pôle intelligence logicielle du Contrat de Plan Etat Région. Il est responsable, pour l'équipe Calligramme, de l'Action de Recherche Coopérative RLT (Ressources Linguistiques pour les TAG). Il est l'organisateur local du colloque IWPT'03 (International Workshop on Parsing Technologies) qui aura lieu à Nancy en 2003.

9.2. Enseignement

- Adam Cichon, en collaboration avec Claude Kirchner, donne le cours *Logique et démonstration automatique* du DEA d'informatique des universités nancéiennes. Il donne également un cours de Logique et Intelligence Artificielle en DESS Compétences Complémentaires. Il est responsable de la licence d'informatique.
- Philippe de Groote, en collaboration avec Didier Galmiche, donne le cours *Calculs pour la modélisation et la preuve* du DEA d'informatique des universités nancéiennes.
- François Lamarche et Philippe de Groote donnent le cours *Sémantique, logique et pragmatique des langues naturelles* du DEA d'informatique des universités nancéiennes.
- Dans le cadre du DEA informatique des universités nancéiennes, Guy Perrier avec Bertrand Gaiffe de l'équipe *Langue et Dialogue* donne un cours d'analyse syntaxique des langues.

9.3. Jurys de thèse

- Adam Cichon a été président du jury de thèse de Q. Nguyen (équipe Protheo). Il a aussi été membre du jury de thèse de E. Deplagne (équipe Protheo).
- François Lamarche était rapporteur et membre du jury de thèse de Richard Moot, Département de Linguistique, Université d'Utrecht, en février.

9.4. Participation à des colloques, séminaires, invitations

- Jérôme Besombes a présenté ses travaux [8] à la Conférence d'Apprentissage CAp 2002 à Orléans en juin 2002.
- Guillaume Bonfante et Jean-Yves Marion ont participé à une formation au logiciel Coq (février 2002).
- Adam Cichon et Jean-Yves Marion ont été invité à un workshop sur la complexité de calcul, à Oberwolfach, Allemagne (avril 2002).
- Philippe de Groote a donné un séminaire au laboratoire de mathématique de l'Université de Savoie (2 mai, 2002).
- Philippe de Groote et Jérôme Besombes ont participé à TAG+6 (Venise, 20-23 mai, 2002) où ils ont présenté leurs travaux [11][9].
- Bruno Guillaume a donné un séminaire au laboratoire de mathématique de l'Université de Savoie (10 janvier, 2002).
- François Lamarche a donné des exposés dans les workshops et séminaires suivants :

  – le workshop organisé à l'occasion de la soutenance de Richard Moot, Utrecht 31 janvier,

  - les « logic and interaction weeks », CIRM (Marseille), 4-21 février, durant lesquelles il a donné un cours de quatre heures intitulé « operads and the geometry of computation »,
  - le workshop « proof, computation and complexity », Tübingen, 8-9 avril,
  - le séminaire CATIA (Montpellier 2), 26 avril,
  - le « workshop on proof theory and computation », Dresde, 3-14 juin, durant lequel il a aussi donné un cours de cinq heures intitulé « operads and the geometry of computation ».
- François Lamarche a également participé à deux séminaires « Catégories du Chevaleret » à Paris (janvier et mai) dans l'un desquels il a donné un exposé.
- Jean-Yves Marion et Jérôme Besombes ont participé à une réunion de l'ARC GRACQ (mars 2002, Paris).
- Jean-Yves Moyen a participé à la conférence FLOC 2002, à Copenhague en juillet.
- Guy Perrier a participé à trois réunions de l'ARC RLT (février, juin et septembre) à Paris.
- Guy Perrier et Btuno Guillaume ont participé à la conférence sur le Traitement Automatique des Langues Naturelles, TALN'02, qui s'est tenue à Nancy du 24 au 27 juin 2002. Guy Perrier y a présenté une communication [13].
- Sylvain Salvati a participé à l'école d'été « Theory and practice of formal proofs », en octobre, à Giens.

10. Bibliographie

Bibliographie de référence

[1] A. CICHON, E. TAHHAN-BITTAR. *Ordinal recursive bounds for Higman's theorem.* in « Theoretical Computer Science », numéro 1-2, volume 201, 1998, pages 63-84.

[2] R. DAVID, B. GUILLAUME. *A λ-calculus with explicit weakening and explicit substitution.* in « Mathematical Structures in Computer Science », numéro 1, volume 11, 2001, pages 169-206.

[3] P. DE GROOTE. *An algebraic correctness criterion for intuitionistic multiplicative proof-nets.* in « Theoretical Computer Science », numéro 1-2, volume 224, 1999, pages 115-134.

[4] D. LEIVANT, J.-Y. MARION. *A characterization of alternating log time by ramified recurrence.* in « Theoretical Computer Science », numéro 1-2, volume 236, 2000, pages 192-208.

[5] G. PERRIER. *Labelled Proof Nets for the Syntax and Semantics of Natural Languages.* in « Logic Journal of the IGPL », numéro 5, volume 7, 1999, pages 629-654.

Articles et chapitres de livre

[6] P. DE GROOTE. *On the Strong Normalisation of Intuitionistic Natural Deduction with Permutation-Conversions.* in « Information and Computation », volume 178, 2002, pages 441-464.

[7] P. DE GROOTE, F. LAMARCHE. *Classical Non-Associative Lambek Calculus.* in « Studia Logica », volume 71, 2002, pages 355-388.

Communications à des congrès, colloques, etc.

[8] J. BESOMBES, J.-Y. MARION. *Apprentissage de langages réguliers d'arbres et applications.* in « Conférence d'Apprentissage CAp 2002, Orléans », juin, 2002.

[9] J. BESOMBES, J.-Y. MARION. *Learning languages from positive examples with dependencies.* in « Proceedings of Sixth International Workshp on Tree Adjoining Grammars and Related Frameworks, Venezia », mai, 2002.

[10] O. BOURNEZ, P. DE NAUROIS, J.-Y. MARION. *Safe Recursion and Calculus over an Arbitrary Structure.* in « Implicit Computational Complexity - ICC'2002, Copenhagen, Denmark », juillet, 2002.

[11] P. DE GROOTE. *Tree-Adjoining Grammars as Abstract Categorial Grammars.* in « TAG+6, Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks », Università di Venezia, pages 145-150, 2002.

[12] F. LAMARCHE. *Multiplicative Linear Logics and Fibrations.* in « Category Theory in computer science - CTCS'2002, Ottawa, Ontario, Canada », série Electronic notes in theoretical computer science, volume 69, R Blute, Ph. Scott, Elsevier North Holland, éditeurs R. BLUTE., novembre, 2002.

[13] G. PERRIER. *Descriptions d'arbres avec polarités : les Grammaires d'Interaction.* in « 9ième Conférence annuelle sur le Traitement Automatique des Langues Naturelles - TALN'02, Nancy, France », juin, 2002, http://www.loria.fr/publications/2002/A02-R-123/A02-R-123.ps.

Rapports de recherche et publications internes

[14] J. BESOMBES, J.-Y. MARION. *Learning regular tree languages with positive samples and queries.* Rapport de recherche, LORIA, octobre, 2002.

[15] G. PERRIER. *Modular and Hierarchical Organization of Lexicalized Grammars expressible with Tree Descriptions.* Rapport de recherche, LORIA, mai, 2002.

Bibliographie générale

[16] D. ANGLUIN. *A note on the number of queries needed to identify regular languages..* in « Information and Computation », volume 51, 1981, pages 76-87.

[17] D. ANGLUIN. *Learning regular sets from queries and counterexamples..* in « Information and Computation », volume 75, 1987, pages 87-106.

[18] Y. BAR-HILLEL. *A quasi-arithmetical notation for syntactic description.* in « Language », volume 29, 1950, pages 47-58.

[19] J. BOS. *Predicate Logic unplugged.* in « 10th Amsterdam Colloquium », éditeurs P. DEKKER, M. STOKHOF., pages 133-142, 1995.

[20] H. COMON, M. DAUCHET, R. GILLERON, F. JACQUEMARD, D. LUGIEZ, S. TISON, M. TOMMASI. *Tree Automata Techniques and Applications.* 1997, http://www.grappa.univ-lille3.fr/tata.

[21] H. CURRY. *Foundations of mathematical logic.* Dover Publications, 1977.

[22] P. DE GROOTE. *Type raising, continuations, and classical logic.* in « Proceedings of the Thirteenth Amsterdam Colloquium », Institute for Logic Language and Computation, Universiteit van Amsterdam, éditeurs R. VAN ROOY, M. STOKHOF., pages 97-101, 2001.

[23] A. DIKOVSKY, L. MODINA. *Dependencies on the other side of the curtain.* in « Traitement Automatique des Langues », numéro 1, volume 41, 2000, pages 67-96.

[24] H. GAIFMAN. *Dependency systems and phrase structure systems.* in « Information and Control », numéro 3, volume 8, 1965, pages 304-337.

[25] 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.

[26] J.-Y. GIRARD. *Linear Logic.* in « Theoretical Computer Science », volume 50, 1987, pages 1-102.

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

[28] M. KANAZAWA. *Learnable classes of Categorial Grammars.* CSLI, 1998.

[29] J. LAMBEK. *The mathematics of sentence structure.* in « Amer. Math. Monthly », volume 65, 1958, pages 154-170.

[30] J. LAMBEK. *On the calculus of syntactic types.* in « Studies of Language and its Mathematical Aspects », Proc. of the 12th Symp. Appl. Math.., pages 166-178, Providence, 1961.

[31] J.-Y. MOYEN. *System Presentation : An Analyser of Rewriting Systems Complexity.* in « Electronic Notes in Theoretical Computer Science », volume 59, Elsevier Science Publishers, éditeurs M. VAN DEN BRAND, R. VERMA., 2001.

[32] S. PINKER. *The language instinct.* Harper, 1994.

[33] Y. SAKAKIBARA. *Learning Context-Free Grammars from Structural Data in Polynomial Time..* in « Theoretical Computer Science », volume 76, 1990, pages 223-242.

[34] Y. SAKAKIBARA. *Efficient learning of context free grammars from positive structural examples.* in « Information and Computation », volume 97, 1992, pages 23-60.