Systèmes déductifs et traitement des langues : un panorama des grammaires catégorielles

Christian Retoré

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles

Christian Retoré

N°3917

Avril 2000

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--3917--FR

Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles

Christian Retoré

Thème 1 — Réseaux et systèmes
Projet Paragraphe

Rapport de recherche n˚3917 — Avril 2000 — 45 pages

Résumé : Depuis les années 80 les grammaires catégorielles connaissent un vif regain d'intérêt. Poursuivant les travaux des années cinquante (voire trente) dans une optique neuve, les recherches récentes les rapprochent des systèmes de typage utilisés en informatique. La linguistique informatique s'est ainsi enrichie d'une approche originale qui néanmoins rejoint d'autres formalismes de traitement automatique des langues plus connus, telles les grammaires d'arbres adjoints, les grammaires de dépendances et certaines grammaires syntagmatiques. On peut se demander quelles sont les raisons d'une telle démarche, quels succès elle a obtenus, quels objectifs étaient, et sont toujours, poursuivis. Nous nous proposons de répondre à ces questions par le panorama que voici, en soulignant l'importance que joue la logique dans les développements récents. Les résultats de base sont explicitement traités tandis que les recherches et réalisations récentes ne sont qu'esquissées. Alors qu'il existe des états de l'art plus spécifiques et plus complets, celui-ci ne nécessite guère de prérequis.

Mots-clé : Grammaires formelles ; logique ; traitement automatique des langues

(Abstract: pto)

Soumis au numéro spécial *Traitement automatique du langage naturel (sous la direction de Daniel Kayser et Bernard Levrat)* de la revue *Technique et Science Informatiques (TSI)*

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Deductive systems and natural language processing: a survey of categorial grammars

Abstract: Since the eighties there has been a revived interest in categorial grammars. Taking ideas that go back to the fifties (and even to the thirties) in a new direction, recent research reveals categorial grammars to be type systems of the sort familiar in computer science. This new trend in computational linguistics is nevertheless tightly related to more common formalisms for natural language processing, like tree adjoining grammars, dependency grammars, and certain phrase-structure grammars. What is the motivation of such an approach? What are its successes? What are its objectives? This survey tries to answer these questions, stressing the important role of logic in recent developments. Basic results are precisely covered while more recent research and insights are only informally presented. While there exists more specific or complete surveys of the state-of-the-art, this one only relies on the common knowledge in computer science.

Key-words: Formal grammar ; logic ; natural language processing

Table des matières

1 Présentation

La linguistique informatique présente deux faces complémentaires : l'une vise à la réalisation d'outils qui permettent un traitement automatique des langues, l'autre cherche à vérifier ou infirmer les théories calculatoires de la langue. Pour traiter de ces aspects, il y a grosso modo deux types d'approches, qui semblent malheureusement inconciliables à l'heure actuelle. L'une traite de grands corpus par des techniques statistiques ou probabilistes ; on peut attribuer la paternité de l'autre, la seule qui nous concernera ici, à Chomsky dans les années 50 : elle propose un traitement symbolique de la langue pour en exhiber les mécanismes calculatoires. [73]

Inscrite dans cette approche calculatoire et symbolique, notre présentation concerne surtout les grammaires catégorielles apparues il y a bien longtemps [7, 8], ainsi que leurs extensions sous la forme générale de ce qu'il est convenu d'appeler les « grammaires de types logiques » [61, 54]. C'est un modèle de la syntaxe des langues, mais qui entretient d'étroits rapports avec la sémantique de Montague [52, 63].

En effet, cette formalisation de la sémantique, qui calcule les formes logiques des énoncés, a très tôt été associée aux grammaires catégorielles (Montague ne présentait-il pas lui-même sa grammaire comme « la » grammaire catégorielle ?). Il est d'usage en linguistique et donc aussi en linguistique informatique de désigner par « sémantique » cette partie de l'analyse linguistique qui s'intéresse à la description (même limitée) des phénomènes de signification. Parmi ceux-ci, les phénomènes de référence ont toujours occupé une place primordiale. En effet, savoir analyser un énoncé, ou un discours entier, ce n'est pas seulement reconnaître sa bonne formation syntaxique ; c'est aussi et surtout donner une représentation de l'énoncé ou du discours qui indique clairement la portée référentielle des expressions, qu'il s'agisse de syntagmes nominaux pleins, de formes pronominales, voire même d'expressions temporelles. Ceci nécessite entre autres une représentation de la portée des quantificateurs et c'est indiscutablement la sémantique de Montague qui est allée le plus loin dans l'approche rigoureuse de ces phénomènes. De plus, l'utilisation d'une logique intentionnelle permet même l'expression de nuances importantes du point de vue du sens dénotationnel, comme par exemple la représentation d'ambiguïtés dans le cas de verbes comme *chercher*. [1]

1. Suivant que l'objet de *chercher* est un objet précis ou hypothétique.

Les grammaires catégorielles entrent dans le cadre d'un lexicalisme radical : comme dans les grammaires d'arbres adjoints lexicalisées voir [35, 1], les règles sont uniformes, données une fois pour toutes et indépendantes de la langue — une sorte de grammaire universelle à la Chomsky [15, 73]. Seule la variation des types que le lexique associe à des entrées lexicales (ou dans les calculs multimodaux de Moortgat [54] les postulats globaux associés à certains opérateurs), expriment les variations d'une langue à l'autre. On notera que les travaux récents de Chomsky [15, 73] optent eux aussi pour des grammaires lexicalisées, et sur ce point au moins les grammaires catégorielles étaient en avance : elles ont toujours prôné l'approche lexicaliste.

La lexicalisation n'est pas qu'un plus esthétique : cette approche est d'un grand intérêt pour l'apprentissage, sujet d'importance primordiale en intelligence artificielle, dans les sciences cognitives et, sous une forme plus spécialisée, en linguistique. Dans le cas qui nous préoccupe, il s'agit d'apprendre la grammaire d'une langue, et ce problème est également connu sous le nom d'inférence grammaticale. [2] D'un point de vue théorique, c'est un pilier des théories chomskiennes [73, 70], qui justifie le postulat d'une grammaire universelle dont les grammaires particulières sont des instances ; du point de vue de la théorie des langages formels, c'est un sujet aux fondements anciens [27], mais en plein essor et renouveau [81], que ce soit pour l'étude du génome, le diagnostic . . . ou le traitement des langues. Les grammaires lexicalisées présentent le grand avantage de ne pas avoir de règles à apprendre, puisque celles-ci sont fixes ; il suffit alors de déterminer l'entrée lexicale associée à un nouveau mot ou au nouvel usage d'un mot déjà présent dans le lexique.

Techniquement, les grammaires catégorielles, depuis leur formulation logique par Lambek en 1958, [41] et plus encore dans les extensions récentes s'apparentent plus à la théorie de la démonstration [24] qu'à la théorie des langages classiques [79], ou qu'aux autres formalismes de traitement automatique des langues [1] ; cela explique en partie leur relatif isolement. L'autre raison de la discrétion des grammaires catégorielles jusqu'aux années 80 est que le calcul de Lambek est trop restrictif, et qu'on ne connaissait pas de liens entre ce genre de calcul logique (linéaire avant l'heure) et les logiques usuelles : classique, intuitionniste et modale. Il a fallu attendre que soient établies des connexions avec des calculs logiques plus standard, c'est-à-dire que la logique s'intéresse à des calculs restreints : des liens ont premiè-

2. Expression à éviter dans le cadre des grammaires catégorielles, car on peut la comprendre comme l'étude des inférences logiques décrivant une grammaire.

rement été établis avec la logique modale, surtout par van Benthem et Moortgat [91, 54], puis avec la logique linéaire inventée par Girard [22, 23, 89], qui entretient d'étroites relations avec les logiques intuitionniste et classique.

Précisons tout de suite que cette approche logique des grammaires catégorielles n'est pas la seule : en poursuivant les calculs de fractions de Bar-Hillel [8] dans le style des calculs de combinateurs, Steedman [86] propose une toute autre approche des grammaires catégorielles qui étend leur capacité générative aux langages légèrement contextuels (mildly context sensitive) [36].

Réciproquement les grammaires catégorielles et la sémantique de Montague ne sont pas le seul contact entre logique et traitement informatique des langues, comme en témoignent le récent *Handbook of Logic and Language* [90] ou les conférences *Logical Aspects of Computational Linguistics* [11]. Ces deux références montrent bien comment les grammaires catégorielles s'inscrivent dans une perspective plus large des liens entre logique, langues et langages ; mais elles constituent aussi en elles-mêmes un domaine d'étude bien vivant, comme en témoigne la présentation plus technique de Lecomte de la journée *Grammaire et théorie de la preuve* [44].

Le panorama que nous proposons est organisé ainsi :

- La première partie reprend des travaux bien connus et simples à présenter assez explicitement : grammaires AB, calcul de Lambek, lien avec la sémantique de Montague.
- La seconde présente informellement les travaux qui ont été réalisés au-delà de ces résultats de base : d'une part les extensions du calcul de Lambek, qu'elles soient dans le cadre des logiques multimodales ou de la logique linéaire.
- La troisième signale les algorithmes et réalisations pour le traitement automatique des langues issus de cette approche : algorithmes d'apprentissage, d'analyse et de génération.
- La dernière montre les liens qu'entretiennent les grammaires catégorielles avec d'autres formalismes plus répandus telles les grammaires d'arbres adjoints, les grammaires de dépendance, et les grammaires minimalistes.

2 Des grammaires catégorielles classiques au calcul de Lambek et à la sémantique de Montague

Les éléments que nous présentons ici se trouvent décrits avec plus de détails dans les chapitres [12, 63] du livre [90].

2.1 Les années 50-60 : grammaires AB et grammaires non contextuelles

Les premières grammaires catégorielles, grammaires AB (d'après Ajdukiewicz [7] et Bar-Hillel [8]) procèdent comme suit. On se donne un ensemble fini P de catégories ou types de base, par exemple sn,n,S (syntagme nominal, nom, phrase) dont l'un joue un rôle particulier : S (l'analogue du non terminal de départ dans une grammaire syntagmatique). Les types ou catégories sont donnés sous la forme : [3]

$$\mathcal{T} ::= P \mid \mathcal{T} \setminus \mathcal{T} \mid \mathcal{T} \,/\, \mathcal{T} \mid \mathcal{T} \bullet \mathcal{T}$$

Un lexique associe à chaque mot m_i un ou plusieurs types $\mathcal{L}(m_i) = \{t_i^1, \ldots, t_i^{k_i}\}$.

On se donne alors des règle de calcul sur les types, qui correspondent à un calcul de fractions non commutatif. Pour toutes catégories x,y :

$$x \bullet (x \setminus y) \longrightarrow y$$
$$(y \,/\, x) \bullet x \longrightarrow y$$

Le langage engendré par cette grammaire (totalement lexicalisée) est défini comme l'ensemble des suites de mots du lexique $m_1 \ldots m_n$ telles que pour chaque m_i il existe un type $t_i \in \mathcal{L}(m_i)$, tel que $t_1 \bullet \cdots \bullet t_n \longrightarrow S$ (où, par abus de langage, on note également $\longrightarrow$ la clôture transitive de $\longrightarrow$).

3. Attention aux notations : dans les grammaires catégorielles combinatoires de Steedman [86], $A \setminus B$ est noté $B \setminus A$ de sorte que type correspondant au but est toujours le premier; le schéma de réduction est alors $A(B \setminus A) \longrightarrow B$. Nous utilisons ici la notation de Bar-Hillel et Lambek dont l'intuition est la suivante : $Y \,/\, X$ et $X \setminus Y$ sont des fractions dans un groupe non commutatif : $Y \,/\, X = YX^{-1}$ et $X \setminus Y = X^{-1}$.

Donnons un exemple minuscule de lexique/grammaire : [4]

Mot	Type(s)
$cosa$	$(S \,/\, (S \,/\, sn))$
$guarda$	$(S \,/\, sv)$
$passare$	$(sv \,/\, sn)$
il	$(sn \,/\, n)$
$treno$	n

La phrase $guarda\ passare\ il\ treno$ (il/elle regarde passer le train) appartient au langage engendré:

$$\begin{array}{l}(S \,/\, sv) \bullet (sv \,/\, sn) \bullet (sn \,/\, n) \bullet n \\ \longrightarrow (S \,/\, sv) \bullet (sv \,/\, sn) \bullet sn \\ \longrightarrow (S \,/\, sv) \bullet sv \\ \longrightarrow S\end{array}$$

tandis que la phrase $cosa\ guarda\ passare$ (que regarde-t-il/elle passer?) n'appartient pas au langage engendré : la séquence $(S \,/\, (S \,/\, sv)) \bullet S \,/\, sv) \bullet (sv \,/\, sn)$ ne contient rien qui se réduise.

Il est facile de montrer que toute grammaire algébrique (ou non contextuelle) sans production vide est faiblement équivalente à une grammaire AB et réciproquement [9] :

– *Des grammaires algébriques aux grammaires AB.* Mettons la grammaire algébrique sous forme normale de Greibach, et donnons au mot (terminal) a le type $(\cdots((X \,/\, X_1) \,/\, X_2)\cdots) \,/\, X_n$ s'il apparaît dans une règle $X \longrightarrow aX_n \ldots X_1$.

  Alors les langages engendrés par la grammaire catégorielle et la grammaire algébrique coïncident. [5]

4. Cet exemple est en italien, car la possible absence de sujet fournit un exemple minimal de l'utilité de la composition, absente des grammaires AB mais présente dans le calcul de Lambek, comme on le verra dans le prochain exemple.

5. En mettant la grammaire sous forme normale de Greibach forte, on peut se limiter à $n \leq 2$, ce qui donne des types de profondeur 1 et de la forme X ou $X \,/\, Y$ ou $(X \,/\, Y) \,/\, Z$. On peut aussi que le symbole $\backslash$ n'a pas été utilisé. Cela signifi e que les grammaires algébriques sont représentables dans un fragment très restreint des grammaires AB.

– *Des grammaires AB aux grammaires algébriques.* Soit $\mathcal{L}$ un lexique décrivant une grammaire catégorielle, notons $\mathtt{Tp}$ l'ensemble des types et des sous-types de tous les types apparaissant dans le lexique — le lexique étant fini, $\mathtt{Tp}$ est fini. Considérons la grammaire algébrique dont $\mathtt{Tp}$ est l'ensemble des non-terminaux et dont les règles sont les suivantes :
  – $T \longrightarrow a$ chaque fois que $T \in \mathcal{L}(a)$
  – $V \longrightarrow (V \,/\, U)\; U$ — avec U,V,$(U \,/\, V) \in \mathtt{Tp}$
  – $V \longrightarrow U\; (U \setminus V)$ — avec U,V,$(U \,/\, V) \in \mathtt{Tp}$

  Cette grammaire algébrique (sous forme normale de Chomsky) engendre le même langage que la grammaire catégorielle initiale.

Ainsi les grammaires AB sont-elles une alternative aux grammaires algébriques, du point de vue des langages engendrés. Mais il s'agit là d'équivalence faible, et la formalisation différente permet déjà d'exprimer différemment le fonctionnement de la grammaire ; de plus elles sont lexicalisées et c'est là une différence essentielle, notamment pour les algorithmes d'apprentissage. On notera qu'une lexicalisation des grammaires algébriques qui préserve les arbres d'analyse nécessite de les étendre, et conduit aux grammaires d'arbres adjoints (TAGs) [1, 35].

Pour ce qui est du sens de leur fonctionnement, il est des règles de calcul qu'on aimerait ajouter, par exemple :

– la montée de type, chère aux grammaires de Montague (cf. ci-après) : $x \longrightarrow (y \,/\, x) \setminus y$;
– la composition, $(x \setminus y) \bullet (y \setminus z) \longrightarrow x \setminus z$, qui aurait permis de reconnaître la phrase $cosa\ guarda\ passare$

Mais où s'arrêter de façon à ce que l'ajout de règles n'induise pas l'incohérence de l'ensemble de ces règles, incohérence qui se traduirait par exemple par l'identification des deux symboles orientés $/$ et $\setminus$ (voir par exemple la discussion dans [53]).

2.2 Le calcul de Lambek (1958) : l'irruption de la logique

En 1958, Joachim Lambek [41] montre qu'on peut formaliser les grammaires catégorielles sous une forme logique (peut-être préfèrerait-il catégorique), et que

toutes les règles de calcul envisagées ci-dessus deviennent dérivables dans son système. Par exemple la règle de simplification des fractions est alors vue comme un *modus ponens* : $A,A\rightarrow B \vdash B$. Le fonctionnement d'une grammaire de Lambek est le suivant :

- Un lexique associe à chaque mot m_i un ou plusieurs types de $\mathcal{T}$ défini précédemment soit $\mathcal{L}(m_i) = \{t_i^1, \ldots, t_i^{k_i}\} \subset \mathcal{P}(T)$ — le lexique est en tout point similaire à celui d'une grammaire AB.
- Le langage engendré par cette grammaire (totalement lexicalisée) est défini comme l'ensemble des suites de mots $m_1 \ldots m_n$ du lexique telles que pour chaque m_i il existe un type $t_i \in \mathcal{L}(m_i)$ de sorte que le calcul de Lambek démontre le séquent
$$t_1, \ldots, t_n \vdash S$$

Avant tout commentaire, donnons immédiatement les règles de ce calcul, à la fois en calcul des séquents puis en déduction naturelle. Les majuscules grecques désignent des suites finies de formules, séparées par des virgules, et ϵ désigne la suite vide.

$$\frac{\Gamma,B,\Gamma' \vdash C \quad \Delta \vdash A}{\Gamma,\Delta,A \setminus B,\Gamma' \vdash C} \setminus_h \qquad \frac{A,\Gamma \vdash C}{\Gamma \vdash A \setminus C} \setminus_i \quad \Gamma \neq \epsilon$$

$$\frac{\Gamma,B,\Gamma' \vdash C \quad \Delta \vdash A}{\Gamma,B / A,\Delta,\Gamma' \vdash C} /_h \qquad \frac{\Gamma,A \vdash C}{\Gamma \vdash C / A} /_i \quad \Gamma \neq \epsilon$$

$$\frac{\Gamma,A,B,\Gamma' \vdash C}{\Gamma,A \bullet B,\Gamma' \vdash C} \bullet_h \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta,\Gamma \vdash A \bullet B} \bullet_i$$

$$\frac{\Gamma \vdash A \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash B}\, coupure \qquad \frac{}{A \vdash A}\, axiome$$

La déduction naturelle ne contient pas les règles $/_h$, $\setminus_h$, $\bullet_h$ ni la $coupure$ mais l'axiome, les règles $/_i$, $\setminus_i$, $\bullet_i$ ainsi que les règles d'élimination suivantes:

$$\frac{\Gamma \vdash A \quad \Delta \vdash A \setminus B}{\Gamma,\Delta \vdash B}\setminus_e \qquad \frac{\Delta \vdash A \bullet B \quad \Gamma,A,B,\Gamma' \vdash C}{\Gamma,\Delta,\Gamma' \vdash C}\bullet_e \qquad \frac{\Delta \vdash B \,/\, A \quad \Gamma \vdash A}{\Delta,\Gamma \vdash B}/_e$$

L'axiome s'applique a priori à toute formule A mais on peut en fait se restreindre à ne l'appliquer qu'à des formules atomiques (des variables propositionnelles) : les axiomes sur des formules composées sont déductibles des axiomes atomiques, et ce sans utiliser la règle de coupure.

La restriction imposée sur les règles $/_i$ et $\setminus_i$ suffit à assurer qu'aucun séquent sans hypothèse (de la forme $\vdash F$) n'est démontrable, c'est-à-dire qu'il n'y a pas de tautologie dans ce calcul. Appelons momentanément L1 le calcul de Lambek sans cette restriction, pour voir qu'elle est bel et bien nécessaire. La signification de $\Gamma \vdash A \setminus B$ est que la chaîne de type Γ doit *nécessairement* être précédée d'une chaîne de type Δ à sa gauche pour donner une chaîne Δ,Γ de type B. Si A est une tautologie de L1, c'est-à-dire si L1 démontre $\vdash A$ alors de $\vdash A$ et $\Gamma \vdash A \setminus B$ on peut déduire $\Gamma \vdash B$, c'est-à-dire que la suite Δ fournie à gauche de Γ est la suite vide. L'exemple linguistique suivant montre qu'on rencontre dans la langue des cas où cette restriction est nécessaire ; le lexique $très : (n \setminus n) \,/\, (n \setminus n)$, $une : sn \,/\, n$, $belle : n \setminus n$, $femme : n$ permet, en l'absence de cette restriction de montrer que $une\ femme\ très$ est un syntagme nominal ; en effet le type adjectif droit $n \setminus n$ demandé par $très$ à sa droite est (comme tous les modifieurs) une tautologie de L1 et peut être fourni par la séquence vide.

Voyons maintenant comment le lexique précédemment donné permet de dériver, dans le calcul de Lambel l'appartenance de $cosa\ guarda\ passare$:

$$\dfrac{(S \,/\, (S \,/\, sv)) \qquad \dfrac{\dfrac{(S \,/\, sv) \vdash (S \,/\, sv) \qquad \dfrac{(sv \,/\, sn) \vdash (sv \,/\, sn) \quad sn \vdash sn}{(sv \,/\, sn),sn \vdash sv}/e}{(S \,/\, sv),(sv \,/\, sn),sn \vdash S}/_i}{(S \,/\, sv),(sv \,/\, sn) \vdash S \,/\, sn}/_e}{(S \,/\, (S \,/\, sv)),(S \,/\, sv),(sv \,/\, sn) \vdash S}$$

Cet exemple, repose sur la composition de $/$. Elle est établie en introduisant un sn fictif, qui est ensuite abstrait par une règle d'introduction : cet sn fictif est assimilable à la trace d'un constituant déplacé dans les théories chomskienne.

On remarque que ce calcul est :

– *Intuitionniste.* Cette logique est intuitionniste, mais ceci est habituel, surtout en informatique : on n'a pas de négation involutive, ni de disjonction, parce qu'on autorise une seule formule à droite du signe « $\vdash$ ».

– *Linéaire* Ce calcul ne comporte aucune des règles dites structurelles de la logique : deux hypothèses ne peuvent être considérées comme une seule hypothèse, et une hypothèse supplémentaire ne peut être ajoutée ; c'est un calcul sensible au ressources : les hypothèses peuvent être vues comme des ressources consommables.

– *Non commutatif.* Autre innovation, les hypothèses ne permutent pas entre elles, c'est-à-dire que le « et », noté « $\bullet$ », de ce calcul n'est pas commutatif : c'est ce qui permet d'avoir deux implications, notées $\backslash$ et $/$, l'une consommant son argument à gauche et l'autre à droite.

On remarque que la conjonction « $\bullet$ » est associative ; elle ne fait qu'exprimer la concaténation des hypothèses : $t_1, \ldots, t_n \vdash u$ est démontrable si et seulement si $t_1 \bullet \cdots \bullet t_n \vdash u$ l'est. On observe également que si $t_1 \bullet \cdots \bullet t_n \longrightarrow u$ dans une grammaire AB, le séquent $t_1 \bullet \cdots \bullet t_n \vdash u$ est démontrable dans le calcul de Lambek — et c'est encore le cas avec les règles supplémentaires telles la montée de type, la composition etc. Néanmoins la réciproque est fausse : le calcul de Lambek n'est pas finiment axiomatisable par des règles de réécriture comme celles précédemment citées : il faut nécessairement adjoindre des règles de la forme : si $A \longrightarrow B$ alors $A' \longrightarrow B'$. Cela est uniquement du à l'exclusion des séquents sans hypothèses : le calcul de Lambek avec séquence vide est finiment axiomatisable. [93, 49]

Entre autres propriétés, mentionnons :

– *La propriété de la sous-formule.* Dans le calcul des séquents la règle de coupure est redondante : sa présence ne permet pas de démontrer de nouveaux séquents ; on peut transformer toute démonstration en une démonstration sans coupure et dans une démonstration sans coupure, il est clair que toute formule de la démonstration est sous-formule d'une des formules du séquent conclusion. Ce résultat n'est plus valide en présence d'axiomes propres (autres que les identités $A \vdash A$), ou du moins doit être restreint. C'est un des avantage du calcul de Lambek que de ne pas nécessiter d'axiomes propres.

  La déduction naturelle possède bien sûr une propriété similaire, toujours en l'absence d'axiomes propres ; toute démonstration se réduit en une démonstration normale dans laquelle on n'élimine pas par une règle $*_e$ un connecteur

$*$ qui vient d'être introduit par une règle $*_i$. Ce processus, analogue à la β réduction du λ-calcul produit aussi des démonstration dans lesquelles toute formule est sous-formule de la conclusion ou d'une des hypothèses.

– *La décidabilité.* Etant donné un séquent à démontrer, ou une phrase à analyser, on peut d'après ce qui précède chercher à le démontrer sans coupure. S'il peut provenir d'une règle autre que la coupure, ce ne peut être que d'un nombre fini de manières, et pour chacune de ces manières on se retrouve avec un ou deux séquents à démontrer, chacun ayant moins de symboles. Le calcul est donc décidable.

– *La complétude vis à vis des modèles monoïdaux.*Considérons un monoïde $(M, \cdot, \epsilon)$ dont on appellera *mots* les éléments — on peut par exemple prendre le monoïde libre engendré par les mots du lexique. Interprétons chaque type atomique p (ou variable propositionnelle, ou catégorie élémentaire) par une partie $[p]$ du monoïde — par exemple par les mots dont p est l'un des types. Interprétons les catégories composées comme suit :

  – $[p \bullet q] = [p] \cdot [q] = \{z \in M \mid \exists a \in [p]\ \exists b \in [q] \quad z = a \cdot b\}$
  – $[p \setminus q] = \{z \in M \mid \forall a \in p \quad a \cdot z \in [q]\}$
  – $[q \,/\, p] = \{z \in M \mid \forall a \in p \quad z \cdot a \in [q]\}$

  On a alors le résultat suivant :

  > Si un séquent $T \vdash U$ est démontrable dans le calcul de Lambek alors quelle que soit l'interprétation des catégories de base $[T] \subseteq [U]$. Réciproquement, étant donné un séquent $T \vdash U$, si pour toute interprétation $[T] \subseteq [U]$ alors $T \vdash U$ est démontrable dans le calcul de Lambek. Cette équivalence entre validité et prouvabilité reste vraie lorsqu'on ne considère que des monoïdes libres, ce qui fait dire à certain que le calcul de Lambek est la logique des monoïdes libres.

Du point de vue de la classe de langages engendrés, qu'a-t-on gagné ? Suivant une conjecture de Chomsky 1963, [14, p. 413] on n'engendre pas de langage non algébrique. C'est effectivement le cas, mais nous ne pouvons, comme nous l'avons fait précédemment, donner les clefs de l'équivalence : c'est un des plus jolis résultats de ces dernières années, établi par Mati Pentus en 1992, [64, 65] et qui repose sur deux ingrédients : un modèle à base groupe libre pour le calcul de Lambek, et un raffinement du théorème d'interpolation établi pour le calcul de Lambek par

Roorda [78] en 90. Par contre du point de vue de la richesse descriptive, saisie par l'équivalence forte, c'est-à-dire entre les arbres de dérivation, les analyses dans le calcul de Lambek donnent des structures d'arbres plus riches [88] : tandis que les arbres de dérivations d'une grammaire algébrique définisse un langage régulier d'arbres, les arbres de déductions normales d'une grammaire de Lambek peuvent constituer un langage non régulier d'arbres (mais ils forment toujours un langage algébrique d'arbres).

2.3 Calcul de Lambek et sémantique de Montague

Jusqu'ici nous nous sommes contentés de l'aspect formel des grammaires catégorielles, mais quel est leur apport pour la linguistique et le traitement automatique des langues ? Outre l'intérêt de la lexicalisation, sur lequel nous reviendrons plus loin, il faut signaler leur interface aisée avec la sémantique de Montague. Ceci vient de ce que les λ-termes simplement typés sont des démonstrations en logique intuitionniste ; en effet l'isomorphisme de Curry-Howard (voir par exemple [24]) montre que les λ-termes simplement typés correspondent aux démonstrations en logique intuitionniste, et réciproquement. Or les analyses syntaxiques sont également des démonstrations, mais dans le calcul de Lambek, et celui-ci se plonge naturellement dans la logique intuitionniste. En effet, en lisant $a \setminus b$ et b / a comme $a \to b$ (l'implication intuitionniste) chaque règle du calcul de Lambek est une règle de la logique intuitionniste.

Afin de voir cela plus en détail, commençons par définir un morphisme des types syntaxiques vers les types sémantiques : ceux-ci sont les formules de la logique minimale (dont le seul connecteur est $\to$, l'implication intuitionniste) construite sur les deux types e (entités) et t (valeurs de vérité).

$(\text{Type syntaxique})^*$	$=$	Type sémantique	
S^*	$=$	t	une phrase est une proposition
sn^*	$=$	e	un syntagme nominal désigne une entité
n^*	$=$	$e \to t$	un nom est un sous-ensemble des entités
$(a \setminus b)^* = (b / a)^*$	$=$	$a^* \to b^*$	étend $(\_)^*$ à tous les types

Pour tout mot m le lexique va également associer, un λ-terme τ_k pour chaque type syntaxique $t_k \in \mathcal{L}(m)$, *de sorte que le type de* τ_k *soit précisément* t_k^*, *le pen-*

dant sémantique de ce type syntaxique. Pour alléger les notation, nous utilisons le parenthésage implicite à droite pour les types, $a{\to}b{\to}c = a{\to}(b{\to}c)$, qui va de paire avec le parenthésage implicite à gauche pour les termes, $T\ U\ V = (T\ U)\ V$.

Il faut bien sûr se donner des constantes pour représenter les opération logiques telles la quantification, la conjonction etc :

Constante	Type
$\exists$	$(e{\to}t){\to}t$
$\forall$	$(e{\to}t){\to}t$
$\wedge$	$t{\to}(t{\to}t)$
$\vee$	$t{\to}(t{\to}t)$
$\supset$	$t{\to}(t{\to}t)$

ainsi que des constantes pour marquer la dénotation des mots du lexique, par exemple :

$regarde$ $\quad \lambda x \lambda y\ (\text{regarde}\ x)\ y \quad x : e,\ y : e, \text{regarde} : e{\to}(e{\to}t)$
« regarde » est un prédicat à deux arguments
$Pierre$ $\quad \lambda P\ (P\ \text{Pierre}) \qquad P : e{\to}t, \text{Pierre} : e$
« Pierre » est vu comme
l'ensemble des propriétés vraies de « Pierre »

Ces constantes peuvent contenir des opérateurs d'intentionnalités, « ˆ » et « ˇ » mais pour simplifier nous les omettons : en effet ces opérateurs ne sont introduits que dans le lexique, ne sont pas perturbés par les règles, et n'influent pas sur l'algorithme d'interprétation des énoncés, tandis qu'une présentation de la logique intentionnelle dépasserait le cadre de ce panorama. Étant données

- *une analyse syntaxique de* $m_1 \ldots m_n$ dans le calcul de Lambek, c'est-à-dire une démonstration $\mathcal{D}$ de $t_1, \ldots, t_m \vdash S$ et
- *la sémantique de chacun des mots* m_1,*... et* m_n, c'est-à-dire des λ-termes $\tau_i : t_i^*$,

on obtient la sémantique de l'énoncé par le simple algorithme suivant que nous décrivons dans le formalisme plus intuitif de la déduction naturelle :

1. Remplacer dans $\mathcal{D}$ tout type syntaxique par son image sémantique ; comme la logique intuitionniste est une extension du calcul de Lambek, on obtient alors une démonstration $\mathcal{D}^*$ en logique intuitionniste de $t_1^*, \ldots, t_n^* \vdash t = S^*$.

2. Cette démonstration en logique intuitionniste par l'isomorphisme de Curry-Howard peut se voir comme un λ-terme $\mathcal{D}^*_\lambda$ simplement typé, qui contient une variable libre x_i de type t^*_i pour chaque mot m_i.
3. Remplacer dans $\mathcal{D}^*_\lambda$, chaque variable x_i par le λ-terme τ_i qui est également de type t^*_i.
4. Réduire le λ-terme obtenu, ce qui fournit la représentation sémantique de l'énoncé analysé (une autre analyse du même énoncé peut fournir une représentation sémantique distincte).

Si l'on préfère on peut réduire les démonstrations plutôt que les λ-termes, ou encore utiliser le calcul des séquents et éliminer les coupures. On peut se demander pourquoi obtient-on un λ-terme qui correspond à une formule du calcul des prédicats. C'est parce que, dans la sémantique des mots seuls les constantes autre que les connecteurs ont des types de la forme $e{\to}e{\to}\cdots{\to}e$ (fonctions) ou $e{\to}e{\to}\cdots{\to}t$ (prédicats) ; il n'est pas bien dur de se persuader que tout λ-terme normal de type t ne possédant que des constantes de ces types-là correspond à une formule du calcul des prédicats.

2.4 Un exemple

Considérons le lexique :

mot	***Type syntaxique t*** ***Type sémantique t**** [*]*** ***Représentation sémantique :* λ*-terme de type t**** [*]
certains	$(S \,/\, (sn \setminus S)) \,/\, n$ $(e{\to}t){\to}((e{\to}t){\to}t)$ $\lambda P{:}e{\to}t\ \lambda Q{:}e{\to}t\ (\exists\ (\lambda x{:}e(\wedge(P\ x)(Q\ x))))$
énoncés	n $e{\to}t$ $\lambda x{:}e(\mathsf{enonce}\ x)$
parlent_de	$sn \setminus (S \,/\, sn)$ $e{\to}(e{\to}t)$ $\lambda x{:}e\ \lambda y{:}e\ ((\mathsf{parler\_de}\ x)y)$
eux-mêmes	$((sn \setminus S) \,/\, sn) \setminus (sn \setminus S)$ $(e{\to}(e{\to}t)){\to}(e{\to}t)$ $\lambda P{:}e{\to}(e{\to}t)\ \lambda x{:}e\ ((P\ x)x)$

Montrons pour commencer que l'énoncé « *Certains énoncés parlent d'eux-mêmes.* » fait bien partie du langage engendré par ce lexique. Pour cela construisons une déduction naturelle de conclusion :

$$(S \,/\, (sn \setminus S)) \,/\, n\ ,\ n\ ,\ sn \setminus (S \,/\, sn)\ ,\ ((sn \setminus S) \,/\, sn) \setminus (sn \setminus S) \vdash S$$

en abrégeant en partie droite les types syntaxiques par $C\ E\ P\ X$:

$$\dfrac{\dfrac{C \vdash (S/(sn\backslash S))/n \quad E \vdash n}{C,E \vdash (S/(sn\backslash S))}\,/_e \quad \dfrac{P \vdash (sn\backslash S)/sn \quad X \vdash ((sn\backslash S)/sn)\backslash(sn\backslash S)}{P,X \vdash (sn\backslash S)}\,\backslash_e}{C,E,P,X \vdash S}\,\backslash_e$$

En utilisant l'homomorphisme des types syntaxique en types sémantiques on obtient la démonstration intuitionniste suivante, où C^*,E^*,P^*,X^* sont des abréviations des types sémantiques associés à C,E,P,X :

$$\dfrac{\dfrac{C^* \vdash (e{\to}t){\to}(e{\to}t){\to}t \quad E^* \vdash e{\to}t}{C^*,E^* \vdash (e{\to}t){\to}t}\to_e \quad \dfrac{P^* \vdash e{\to}e{\to}t \quad X^* \vdash (e{\to}e{\to}t){\to}e{\to}t}{P^*,X^* \vdash e{\to}t}\to_e}{C^*,E^*,P^*,X^* \vdash t}\to_e$$

Le λ-terme codant cette démonstration est tout simplement ((c e) (x p)) de type t où c,e,x,p sont des variables de types respectifs C^*,E^*,X^*,P^*.

Remplaçons ces variables par les λ-termes de mêmes types associés aux mots par le lexique — en omettant les types à l'intérieur des λ-termes, qui figurent dans le lexique ci-dessus. On obtient le λ-terme de type t suivant (écrit sur les deux premières lignes), que l'on réduit :

$$\begin{array}{c}\Big((\lambda P\ \lambda Q\ (\exists\ (\lambda x(\wedge(P\ x)(Q\ x)))))(\lambda x(\texttt{enonce}\ x))\Big)\\ \Big((\lambda P\ \lambda x\ ((P\ x)x))(\lambda x\ \lambda y\ ((\texttt{parler\_de}\ x)y))\Big)\end{array}$$

$$\downarrow \beta$$

$$(\lambda Q\ (\exists\ (\lambda x(\wedge(\texttt{enonce}\ x)(Q\ x)))))(\lambda x\ ((\texttt{parler\_de}\ x)x))$$

$$\downarrow \beta$$

$$(\exists\ (\lambda x(\wedge(\texttt{enonce}\ x)((\texttt{parler\_de}\ x)x))))$$

Ce terme réduit, en se souvenant que le x est de ce terme est de type e, représente la formule du calcul des prédicats suivante :

$$\exists x : e\ (\texttt{enonce}(x) \wedge \texttt{parler\_de}(x,x))$$

C'est bien la représentation sémantique de l'énoncé précédemment analysé.

2.5 Limites et atouts de ce formalisme de base

Ce qui précède est formellement satisfaisant, mais ne suffit pas à rendre compte de bon nombre de phénomènes linguistiques.

Pour commencer nous n'avons pas pris en compte les questions morphologiques telles l'accord : elles sont bien sûr prises en compte par des traits (*features*). On peut se contenter, à la différence d'autres formalismes, de traits atomiques, et cela assure la décidabilité de la recherche de démonstration, et donc de l'analyse — du point de vue formel, ces traits sont vus comme des variables et constantes du premier ordre.

Cette question réglée, on se rend compte que ce formalisme souffre d'autres limites bien plus gênantes :

1. *Le problème des ambiguïtés fallacieuses.*[6] Les démonstrations du calcul de Lambek induisent des parenthésages de l'énoncé analysé, et on aimerait qu'ils représentent ses structures en constituants. Une première méthode pour associé un parenthésage à une analyse consiste à structurer les démonstrations comme suit, ce qui est toujours possible : la démonstration commence par déduire dans le calcul de Lambek de chaque type t du lexique un type t' puis compose ces types t' selon les règles des grammaires AB, ce qui donne un parenthésage de l'énoncé analysé. Suivant cette méthode naturelle, tous les parenthésages sont possibles, ce qui ne donne pas de notion de structure en constituants. C'est l'une des critiques les plus vives qui soit pour les grammaires catégorielles. Heureusement Tiede [88] a considéré le parenthésage induit par les déductions naturelles normales du calcul de Lambek. Cette fois tous les parenthésages ne sont plus possibles, mais il en reste encore trop, notamment à cause des commutations de règles, et de l'associativité. Comme on le verra au paragraphe suivant, aussi bien les grammaires multimodales que les grammaires en réseaux de démonstration évitent cet écueil (voir pages 15 et 26).
2. *L'absence de certaines constructions syntaxiques.* Comme seuls les langages algébriques sont engendrés, on ne peut rendre compte, par exemple, des *dépendances croisées* (telles les complétives du néerlandais). Ces constructions sont un problème pour nombre de grammaires, mais il est des constructions bien plus simples qui échappent aussi au calcul de Lambek telles les *constituants discontinus* (la négation en français) et certaines *extractions* (la topicalisation en français, certaines relatives objets) échappent aux grammaires de Lambek. Ces questions peuvent être traitées par les extensions présentées à la

6. Appelées « *spurious ambiguities* » en anglais.

section suivante, que ce soient les grammaires multimodales (page 15) ou les grammaires en réseaux (page 26).

Par contre, au vu de la présentation qui précède, ces grammaires possèdent des atouts qu'il faut bien sûr préserver lorsqu'on en construit des extensions :

- *La relative efficacité de l'analyse syntaxique.* Puisque ce genre d'approche ramène l'analyse syntaxique à de la démonstration automatique dans des logiques sous structurelles, on peut espérer des algorithmes relativement rapides — pour des algorithmes produisant une analyse qui permet un traitement sémantique.
- *Un certain réalisme psycholinguistique.* Si la structure des démonstrations reflète bien la structure syntaxique, alors on peut s'attendre à des mesures formelles qui correspondent aux résultats empiriques sur la manière dont les humains analysent les énoncés.
- *Des algorithmes d'apprentissage, d'inférence de grammaire.* Dans les grammaires lexicalisées dont nous nous occupons, l'apprentissage est a priori plus simple. Qu'en est-il dans la pratique ? On se ramène à des algorithmes d'unification de types, très rapides comme on le verra après. On pourra aussi s'aider des représentations sémantiques des énoncés, puisque celles-ci sont proches de la structure syntaxique.
- *Des algorithmes de génération d'énoncés.* Si l'interface entre syntaxe et sémantique est aisée, on peut espérer reconstruire une structure syntaxique à partir d'une représentation sémantique. Bien sûr le choix des mots nécessite un traitement à part mais on peut déjà obtenir de bons résultats.

La section suivante décrit ce qui a été fait pour préserver ces atouts mais dépasser les limites mentionnées ci-dessus : c'est bien sûr en tirant parti des techniques que la logique fournit aux grammaires catégorielles qu'on y arrive.

3 Développements et extensions du calcul de Lambek

Dans les années quatre-vingt les avancées de la logique ont permis d'enrichir la capacité descriptive des grammaires de Lambek et aussi de concevoir des algorithmes d'analyses, de génération et d'apprentissage. Lecomte [44] donne une description plus conséquente et plus technique des résultats établis avant 96.

3.1 Calcul multimodaux

Pour répondre aux exigences de la modélisation linguistique, et dépasser le cadre que nous avons décrit jusqu'ici, des calculs modaux ont été introduits à peu près à la même époque et indépendamment par Moortgat [53, 54] d'un côté et par Morrill [61] et Hepple [28] de l'autre. Le système minimal qu'ils considèrent est le calcul de Lambek non associatif [42]. Ce système procède comme celui que nous avons donné précédemment, si ce n'est que les hypothèses sont munies d'une structure d'arbre binaire, décrite par des parenthèses $(\_,\_)$: les règles binaires précisent bien sûr comment les deux arbres binaires d'hypothèses se combinent en un seul arbre binaire. Donnons les règles pour $\bullet$ et $\backslash$, celles pour $/$ se déduisant par symétrie de celles pour $\backslash$:

$$\frac{(A,\Gamma) \vdash C}{\Gamma \vdash A \setminus C}\setminus_i \quad \frac{\Gamma[B] \vdash C \quad \Delta \vdash A}{\Gamma[(\Delta, A \setminus B)] \vdash C}\setminus_h \quad \frac{\Gamma[(A,B)] \vdash C}{\Gamma[A \bullet B] \vdash C}\bullet_h \quad \frac{\Delta \vdash A \quad \Gamma \vdash B}{(\Delta,\Gamma) \vdash A \bullet B}\bullet_i$$

Il est alors impossible, dans un tel calcul de montrer la règle de composition $(A \setminus B) \bullet (B \setminus C) \vdash A \setminus C$ ni l'associativité de « $\bullet$ », $A \bullet (B \bullet C) \vdash (A \bullet B) \bullet C$ — d'où le nom calcul non associatif. L'idée de base est d'étendre ce calcul minimal dans deux directions :

- En traitant dans un même calcul logique de plusieurs familles de connecteurs $(\backslash^i, /^i, \bullet^i)$ en même temps et en introduisant à chaque fois l'opération correspondante sur les contextes, c'est-à-dire des paires de parenthèses $(\_,\_)^i$, ce qui revient à avoir un arbre binaire d'hypothèses dont les nœuds internes sont étiquetés par l'un des indices ^{i}.
- En incluant également des modalités, $\diamond, \square^{\downarrow}$, ou connecteurs unaires, définis par des règles qui rappellent celles de la logique modale, d'où les notations.

Les divers modes de composition sont introduits pour avoir, par exemple, une composition associative, mais aussi une qui ne le soit pas. L'intérêt des modalités est que celles-ci peuvent dépendre du contexte : il est possible, par exemple d'exiger qu'une formule ou un contexte soit précédé d'une modalité pour pouvoir lui appliquer une règle. On pourra avoir, par exemple, des règles comme :

$$\frac{\Gamma[(\Delta_1,(\Delta_2)^\diamond)] \vdash A}{\Gamma[(\Delta_1,\Delta_2)^\diamond] \vdash A} \quad \frac{\Gamma[(\Delta_1,(\Delta_2,\Delta_3))] \vdash A}{\Gamma[((\Delta_1,\Delta_2),\Delta_3))] \vdash A}$$ dès qu'un Δ_i est de la forme $(\_)^\diamond$

Le plus intéressant est en fait les liens qu'on peut introduire entre les modalités ou les connecteurs binaires : ceux-ci sont soient décrits dans le calcul des séquents en imposant des restrictions sur la forme des séquents prémisses, soit par des ensembles de postulats ou d'axiomes, peut-être plus intuitifs que des règles du calcul des séquents. Ces modalités peuvent se voir comme des opérateurs de contrôle qui permettent de retrouver un fonctionnement plus libre (par exemple associatif) à partir d'un fonctionnement plus contraint (par exemple non associatif) : cela ressemble aux modalités de la logique linéaire (paragraphe 3.2 ci-après) qui permettent de contrôler la duplication des hypothèses qui est prohibée dans le système de base. Moortgat et d'autres ont alors montré des théorèmes de plongements fidèles des systèmes contraints dans les systèmes moins contraints à l'aide de ces modalités structurelles. [39] La base de ce calcul étant non associative, on évite par la même les ambiguïtés fallacieuses évoquées dans le point 1 de la page 19 mais les postulats impliquant un déplacement ou un changement de parenthésage rendent souvent difficile la lecture de la structure en constituants.

Il est à noter que ce genre de calcul conserve la propriété de la sous-formule ; en présence de postulats (d'axiomes propres) la version restreinte de la propriété de la sous-formule peut suffire (et suffit dans toutes les modélisations linguistiques faites jusqu'ici) à garantir la décidabilité de l'analyse syntaxique.

Le choix de tel ou tel ensemble de postulats pour une langue rappelle bien sûr les « paramètres » qui définissent la spécificité d'une langue dans les théories chomskiennes [73]. On arrive ainsi dépasser les limites décrites en 2 en page 19 en rendant compte de phénomènes linguistiques subtils comme :

- l'extraction médiane, c'est-à-dire l'extraction d'un constituant situé à l'intérieur d'un constituant déjà construit, ce qui se produit notamment lors de la construction d'une relative dont l'élément extrait n'est ni le premier ni le dernier de la clause (*Spirou a vu Fantasio hier. – Fantasio*$_i$ *que [Spirou a vu* t_i *hier]* ...)
- les îlots, dont on ne peut extraire un constituant, (*La mouette que Gaston aime et Lebrac déteste* t_i... - **La mouette que Gaston aime Julie et Lebrac déteste* t_i...)
- les dépendances croisées des complétives en néerlandais.

Il est à noter que la construction de la sémantique de Montague donnée pour le calcul de Lambek s'étend parfaitement à ces systèmes enrichis, sans étendre le λ-calcul (sans quoi celui-ci ne correspondrait plus au calcul des prédicats).

Morrill [61] propose une vision légèrement différente de ces mêmes constructions : elles ont toutes une trace au premier ordre et les structures sémantiques (de Montague), syntaxiques et prosodiques sont traitées simultanément.

3.2 La logique linéaire et les réseaux de démonstration

Comme on l'a dit, le calcul de Lambek est le fragment intuitionniste de la logique linéaire non commutative. La logique linéaire [22, 23, 89] permet en effet de considérer des variantes non commutatives, introduite et étudiées par Abrusci [3] ; en fait il faut même être dans le cadre de la logique linéaire pour obtenir une « bonne » logique non commutative.

L'une des particularités de la logique linéaire (en l'absence des constantes), est que le calcul intuitionniste s'obtient par une simple restriction de langage, voir par exemple [74]. On peut ainsi formuler le calcul de Lambek comme un calcul classique, avec des séquents sans formules à gauche du signe « $\vdash$ », puisqu'on dispose d'une négation involutive notée $(\_)^{\perp}$ qui satisfait les lois de de Morgan. Conséquemment les formules sont définies à partir de la négation, de la conjonction et de la disjonction notée $\wp$; on a alors $A \setminus B = A^{\perp} \wp B$ et $B / A = B \wp A^{\perp}$. Les formules classiques $\mathcal{F}$, les formules intuitionnistes positives $\mathcal{P}$ et les formules intuitionnistes négatives $\mathcal{N}$ sont définies comme suit :

$$\begin{array}{rcl|l|l|l} \mathcal{F} & = & P & P^{\perp} & \mathcal{F} \wp \mathcal{F} & \mathcal{F} \bullet \mathcal{F} \\ \mathcal{P} & = & P & \mathcal{P} \bullet \mathcal{P} & \mathcal{N} \wp \mathcal{P} & \mathcal{P} \wp \mathcal{N} \\ \mathcal{N} & = & P^{\perp} & \mathcal{N} \wp \mathcal{N} & \mathcal{P} \bullet \mathcal{N} & \mathcal{N} \wp \mathcal{P} \end{array}$$

Les formules de $\mathcal{P}$ sont des formules du calcul de Lambek, exprimables avec $\bullet, \setminus, /$, et les formules de $\mathcal{N}$ sont les négations des formules du calcul de Lambek. Si l'on applique les règles classiques (formulées avec des séquents unilatères), uniquement lorsque toutes les formules du séquent conclusion sont intuitionnistes, on obtient le calcul des séquents linéaires intuitionniste : une formule est positive (la conclusion du séquent intuitionniste), et toutes les autres sont négatives (les hypothèses). Pour plus de détails, on pourra consulter [74, 40].

La syntaxe naturelle où exprimer les démonstrations de la logique linéaire est celle des réseaux de démonstration. [22, 23] Cette syntaxe identifie nombre de démonstrations que le calcul des séquents distingue à tort : par exemple, certaines démonstrations ne se distinguent que par l'ordre d'application de certaines règles, tandis que celles-ci permutent. Si, comme le suggèrent les grammaires catégorielles, la structure syntaxique d'un énoncé est effectivement celle d'une démonstration, il devrait y avoir un gain à les représenter comme des réseaux de démonstration. C'est effectivement le cas.

Qu'est ce qu'un réseau de démonstration ? Il représente une démonstration d'une formule F comme un graphe dont la structure qu'on forme ainsi :

- on part de l'arbre des sous-formules de F
- on ajoute des arêtes entre feuilles qui sont la négation l'une de l'autre. Ces arêtes représentent les axiomes $\vdash a, a^{\perp}$ qui commencent les démonstrations : elles sont deux à deux sans sommet commun, et couvrent toutes les feuilles de l'arbre.

Bien sûr toute telle structure n'est pas une démonstration, sinon $a \bullet a^{\perp}$ (soit $a \wedge \neg a$ avec des notations plus standard) serait démontrable. Seuls les graphes satisfaisant un critère sont des démonstrations : il existe maintenant un grand choix de tels critères, par exemple [18, 5] pour ne citer que des travaux récents. Ici, nous ne souhaitons pas entrer dans les détails, mais donnons tout de même le plus simple : *tout cycle du graphe contient les deux arêtes d'un même branchement* $\wp$. Avec des critères complets, on peut totalement se passer du calcul des séquents ou de la déduction naturelle, et donc décrire ainsi l'analyse d'un énoncé : il s'agit alors d'algorithmes sur les graphes qui correspondent aux démonstrations ou analyses syntaxiques. [74]

Ne serait-ce que pour le calcul de Lambek usuel, cette représentation a déjà porté ses fruits :

- *Psycholinguistique, performance d'un locuteur.* M. Johnson [32] a étudié à l'aide des réseaux de démonstration la complexité des énoncés comportant des relatives imbriquées. En effet les axiomes entre deux conclusions correspondent précisément aux dépendances incomplètes qui jusqu'ici étaient une notion assez informelle. On obtient une mesure qui corrobore les résultats expérimentaux ; celle-ci prend en compte plusieurs paramètres : le nombre

d'axiomes entre deux mots dans l'analyse, celles de même nature, et la profondeur de la catégorie attendue dans la formule d'arrivée. G. Morrill [59] a mené plus loin ce type d'approche, en traitant par des techniques similaires d'autres phénomènes : la préférence gauche-droite pour la portée des quantificateurs, ou pour le rattachement le plus bas dans la structure en constituants, etc. Il fait cette fois la distinction entre les axiomes de a vers $a^{\perp}$ (qui correspondent à l'attente d'un constituant) et ceux de $a^{\perp}$ vers a qui fournissent un constituant et ne gênent donc pas l'analyse.

– *Interprétation sémantique.* L'algorithme que nous avons décrit en 2.3 a été explicité par Ph. de Groote et C. Retoré dans les réseaux de démonstration [20], puisque ceux-ci décrivent toute la logique linéaire qui inclut la logique intuitionniste. Le codage de la logique intuitionniste ou des λ-termes dans la logique linéaire permet d'éviter le plus possible les étapes de normalisation d'un coût algorithmique élevé.

– *Génération à partir de représentations sémantiques.* Parmi les critères de correction de démonstrations, certains reposent sur la correspondance entre λ-termes et réseaux (les uns et les autres étant des notations pour les démonstrations). Utilisant ce principe, G. Morrill et J.-M. Merenciano [50] on montré comment reconstruire dans des cas restreints, sans constantes logiques, un réseau (une analyse syntaxique, et donc un énoncé) dont l'interprétation soit celle choisie. S. Pogodalla a étendu leur technique pour y incorporer les constantes logiques, en exploitant [20].

Dans un cadre différent mais qui s'inspire des grammaires catégorielles, Ph. Blache en a fait une utilisation originale comme structure de pré-analyse pour lever des ambiguïtés, pour tout formalisme, mais plus particulièrement pour les TAGs ou les grammaires catégorielles [10].

Cette syntaxe des réseaux a récemment été adaptée aux calculs qui incluent plusieurs mode de compositions, c'est-à-dire plusieurs familles de connecteurs de même nature. D'une part pour les extensions de la logique linéaire qui comportent et des connecteurs commutatifs et des connecteurs non commutatifs [80, 5], et pour les calculs multimodaux [58].

Un autre intérêt de la logique linéaire est qu'elle permet d'étendre les grammaires de Lambek dont on a vu qu'elles étaient trop limitées. Dans cette optique, Lecomte et Retoré [46, 47] ont proposé des grammaires où le lexique associe non

pas une simple formule mais une démonstration partielle, dans une extension de la logique linéaire qui inclut un connecteur non commutatif [76]. L'analyse syntaxique consiste alors à assembler les réseaux de démonstrations incomplets que le lexique associe aux mots en un réseau de démonstration complet, par deux types de branchements : par axiomes (analogue de la substitution dans les grammaires usuelles) et par coupure (qui ressemble à l'adjonction dans les TAGs). Ce modèle reconnaît un ordre partiel sur les mots de la phrase dont toutes les linéarisations sont considérées comme correctes. On peut ainsi traiter des constituants discontinus, de l'ordre des mots relativement libre, et des clitiques ce qui répond aux questions soulevés au point 2 de la page 19. Les ambiguïtés fallacieuses, problème soulevé au point 1 de la page 19 disparaissent, puisqu'on spécifie le parenthésage en associant un réseau partiel à chaque mot (un peu comme dans les LTAGs).

G. Perrier dans [66] a aussi développé un modèle où le lexique associe une démonstration partielle à chaque mot, mais nous le présenterons dans le paragraphe qui suit.

4 Algorithmes et réalisations pratiques

Avant de présenter ces réalisations pratiques, il faut dire qu'il ne s'agit pour l'instant que de prototypes de type académique, dont on espère qu'ils seront un jour intégrés dans des applications réelles. Cette discrétion de l'utilisation pratique des grammaires catégorielles a plusieurs causes, essentiellement sociologiques :

- Cette approche n'a qu'une dizaine d'années ; même lorsqu'un nombre conséquent de chercheurs et ingénieurs se consacrent au développement de logiciel basés sur un modèle, il faut une vingtaine d'années avant que de réelles applications existent. On notera d'ailleurs que les formalismes similaires les plus utilisés dans les applications réelles sont anciens : grammaires algébriques et extensions diverses telles les grammaires de clauses définies (DCG).
- Étant récente, la communauté des grammaires catégorielles est relativement petite, mais, et c'est là un plus lourd handicap, les techniques qu'elles utilisent et en particulier l'étude des démonstrations formelles s'apparentent plus à la programmation fonctionnelle typée qu'aux autres formalismes de traitement des langues — qui sont plutôt liés à la programmation logique et l'unification, où la logique si elle est utilisée, l'est comme langage de description.

– Finalement l'intérêt pour le traitement des langues est de plus en plus motivé par la recherche d'informations sur internet. On cherche alors des analyses incomplètes mais rapides (analyseurs robustes, analyseurs de surface), et pour ce genre de choses les techniques statistiques ou les grammaires de dépendance sont bien plus efficaces.

Quelles applications peuvent tirer parti du formalisme catégoriel? Ce formalisme est essentiellement dédié à la syntaxe, et va de paire avec une sémantique à la Montague qui consiste essentiellement en un calcul de références, et de portée des quantifications. Les grammaires catégorielles semblent donc naturellement destinées à la conception de modules de génération et d'analyse, qui interviennent par exemple dans les logiciels d'aide à la traduction. [7]

En aval de ces modules, se trouvent les algorithmes d'apprentissage, qui assignent automatiquement des types aux mots à partir d'énoncés supposés corrects. En définissant le lexique, on définit en fait une grammaire catégorielle ; cette tâche habituellement ardue est très sensible à la représentation de la grammaire et elle est particulièrement simplifiée par le formalisme catégoriel. On notera que ces algorithmes d'apprentissage s'appliquent à d'autre domaines que le traitement des langues, tels la génomique ou l'extraction d'information. Ce dernier type d'algorithme et les applications qui en résultent occupent une place à part. Non seulement il se situe en aval des problèmes classiques d'analyse et de génération, mais il s'en distingue aussi techniquement : tandis que l'analyse ou la génération souffrent plutôt du trop faible pouvoir d'expression des grammaires catégorielles, [8] l'apprentissage nécessite de considérablement restreindre la classe de langages décrite.

4.1 Apprentissage (inférence grammaticale)

Pour des raisons cognitives, on peut supposer que l'apprentissage par l'enfant de sa grammaire se fait sur la base d'exemples positifs exclusivement [26, 70, 69], et en petit nombre comparé à la complexité de la grammaire qu'il acquiert, du reste

7. Ces modules doivent nécessairement prendre la morphologie en compte, mais comme on l'a déjà dit, ce n'est pas un problème.

8. Par exemple HPSG permet de décrire tous les langages récursivement énumérables. Mais est-ce un gain, puisque les linguistes s'entendent pour dire que les langues humaines sont à peine au dessus des langages algébriques, et qu'en général l'appartenance d'une phrase à un langage récursivement énumérable est indécidable.

rapidement. C'est pour cela qu'on en vient à postuler l'existence d'une grammaire universelle innée qui se spécialise à celle de la langue à apprendre par instanciation de paramètres. [73, 15]

L'identification à la limite en théorie des langages formels, introduit par Gold en 1967 [27] propose un modèle informatique de l'apprentissage naturel. On se donne une classe de grammaires (ou de langages) T, et une énumération des énoncés d'un langage L de T : e_1,e_2, ... Il s'agit de définir un algorithme qui associe à tout ensemble fini d'énoncés E une grammaire $G(E)$ de T engendrant les énoncés de E, de sorte que la suite de grammaires $G_n = G(\{e_1, \ldots, e_n\})$ soit constamment égale à une grammaire produisant précisément L à partir d'un certain rang n.

L'inférence grammaticale à partir d'exemples positifs est une tâche ardue : même pour les langages réguliers il y a de sévères limitations [27]. Aussi ne peut-on pas espérer des résultats mirobolants. Néanmoins pour les grammaires catégorielles de base (qui engendrent des langages algébriques), il existe de bons algorithmes d'apprentissage, efficace et qui permettent l'identification à la limite ; ces algorithmes, présentés dans l'ouvrage récent de M. Kanazawa [37], confèrent aux grammaires catégorielles un remarquable potentiel applicatif pour l'apprentissage.

Le cas le plus simple de ce type d'algorithme est le cas des grammaires AB dites « rigides » où chaque mot se voit attribuer un seul type (quitte à dire, par exemple que « le » article et « le » pronom son deux mots différents, c'est-à-dire à fournir des exemples indexés). On suppose que les exemples positifs sont structurés, c'est-à-dire que l'on a un arbre représentant la structure en constituants, et, pour chaque composition de deux constituants, qu'on sait lequel est la fonction (de type $X \setminus Y$ ou Y / X), et lequel est l'argument (de type X). Dans ce cadre restreint, W. Buszkowski et G. Penn [13] ont montré qu'un algorithme d'unification sur les types permettait de construire la grammaire (le lexique) le plus général engendrant les énoncés proposés comme exemple positifs, et ce en temps $n \log(n)$.

M. Kanazawa a étendu cet algorithme au cas ou le nombre de types que le lexique associe à un mot est borné par un entier k : bien qu'utilisable en pratique, cela introduit un facteur exponentiel dans le pire cas. Si l'on souhaite ne travailler qu'à partir de suites de mots alors il faut envisager toutes les manières possibles de structurer les énoncés, et comme on s'en doute, cela introduit également un facteur exponentiel.

Néanmoins ces techniques d'apprentissage sont bonnes, car on apprend des langages *algébriques* : pour ces derniers, les techniques usuelles basées sur des gram-

maires syntagmatiques et les automates à pile nécessitent également des exemples structurés [81], et en fait fonctionnent assez mal. De plus les hypothèses faites sur les exemples positifs dans les grammaires catégorielles sont très vraisemblables d'un point de vue cognitif — la notion de « tête » ou de « fonction » dans un syntagme étant vite perçue par l'enfant. Ces techniques d'apprentissage basées sur l'unification sont en fait génériques et permettent de traiter de nombreuses questions d'apprentissage comme l'a montré J. Nicolas. [62]

Récemment I. Tellier a eu la bonne idée de supposer connue la sémantique des énoncés, ce qui est plus que raisonnable d'un point de vue psycholinguistique. [9] En exploitant la correspondance entre sémantique de Montague et analyse syntaxique dans les grammaires catégorielles on obtient alors des algorithmes d'apprentissage plus performants et remarquablement réalistes d'un point de vue cognitif, même s'ils restent encore à préciser et développer. [87]

Les techniques d'apprentissage de langages s'appliquent également à la construction de grammaire du génome, et à l'extraction de données. Pour le premier point il n'y a pas encore d'utilisation des grammaires catégorielles, mais comme le génome contient des structures parenthésées que la description usuelle des langages algébriques ne permet pas d'apprendre, ce pourrait être un succès pour les grammaires catégorielles. Pour l'extraction de données, la société *Syllogic* projette d'utiliser les algorithmes d'apprentissage des grammaires catégorielles pour apprendre des données structurées, en se basant sur les travaux de P. Adriaans [6].

4.2 Analyse

Les algorithmes d'analyse syntaxique produisant des représentations sémantiques à la Montague ont été implantés : ils consistent essentiellement en un démonstrateur automatique pour la logique considérée et en de petits lexiques. On en trouve plusieurs versions en libre accès sur internet, par exemple Natural Deduction CG parser (Bob Carpenter, Pittsburg), LexGram [10] (Esther König, Stuttgart), CATLOG et MCGTOOLS (Glyn Morrill et Xavier Llore, Barcelone). Nous allons ici brièvement décrire deux récentes implémentations de ce type de formalismes qui

9. Par exemple les enfants n'apprennent pas une langue sur la seule base de la télévision où pourtant une partie de la sémantique est visible : il faut que le contexte sémantique et pragmatique soit particulièrement éclairant pour les débuts de l'apprentissage.[26]

10. Basé sur CUF et qui fonctionne aussi pour HPSG

correspondent aux deux types d'extensions précédemment décrites : grammaires catégorielles multimodales, et grammaires de réseaux de démonstration.

4.2.1 Grail

L'analyseur le plus complet développé à ce jour pour les grammaires catégorielles est assurément celui d'Utrecht réalisé par R. Moot [56, 57] ; il traite du néerlandais, de l'anglais, de l'italien et du français dans les systèmes multimodaux de M. Moortgat. Le cœur est un démonstrateur automatique pour les logiques catégorielles multimodales(cf. paragraphe 3.1) ; ce programme de 8000 lignes mêle SICStus ProLog et TclTk en utilisant la bibliothèque TclTk fournie avec SICStus ProLog. Son utilisation ne nécessite pas de connaître ProLog et l'interface écrite en TclTk est agréable : une fenêtre permet d'accéder au lexique (modifiable), une aux postulats (modifiables), et les diverses représentations des analyses syntaxiques s'affichent chacune dans une fenêtre de la manière la plus lisible possible.

Lorsqu'il analyse une phrase, ce logiciel fournit plusieurs représentations de son analyse, c'est-à-dire de la démonstration dans la logique catégorielle multimodale : déduction naturelle, calcul des séquents, réseau de démonstration (et produit le fichier LaTeX permettant de les imprimer). À partir de l'analyse syntaxique obtenue, le logiciel construit par un étiquetage de la démonstration la représentation sémantique de l'énoncé.

Pour l'italien et le français Grail inclut des ensembles de postulats modaux pour traiter des pronoms clitiques ; ce module conçu par E. Kraak [38] est assez impressionnant car les constructions sont réellement décrites (par des déplacements et non par la liste de tous les cas possibles) ; cela permet de traiter correctement l'impossibilité de coordonner des clitiques et la montée (ou non) des clitiques suivant les auxiliaires modaux — et par exemple de reconnaître *Je peux la réparer. Je la fais réparer. Marie le laisse la manger.* et aucune autre permutation de ces phrases. Bien sûr les assignations de types sont complexes et font apparaître jusqu'à huit modalités différentes, et les postulats régissant le fonctionnement de ces opérateurs ralentissent l'analyseur syntaxique. Mais les résultats sont corrects et ne reposent pas sur une liste exhaustive des configurations possibles.

4.2.2 Grammaires d'interaction et programmation logique par contraintes

À partir des réseaux de démonstration de la logique linéaire intuitionniste commutative, Guy Perrier (Nancy) a développé un formalisme grammatical à partir de l'idée suivante : utiliser la sensibilité aux ressources de la logique linéaire pour représenter de façon uniforme (à l'aide d'un réseau de démonstration) les dépendances entre constituants syntaxiques d'une phrase, qu'elles soient lointaines ou locales [68].

La spécificité de la logique linéaire intuitionniste permet de donner une représentation plus compacte des réseaux de démonstration à l'aide d'arbres polarisés et sous-spécifiés. La polarisation ajoute à chaque nœud une polarité -1, 0 ou +1 et la sous-spécification sert à indiquer qu'un nœud en domine un autre — en ne précisant pas d'emblée le chemin qui permettra de réaliser cette domination.

Dans ce formalisme issu des réseaux de démonstration, les arbres polarisés et sous-spécifiés sont les structures syntaxiques de base et la composition syntaxique est réalisée par branchement de nœuds duaux [67]. Ce formalisme est actuellement raffiné, en descendant les polarités du niveau des constituants (les nœuds des arbres syntaxiques) au niveau des traits qui sont attachés à ces constituants (sous forme de matrices). La composition syntaxique est alors réalisée sous forme de superposition partielle d'arbres contrôlée par l'interaction « électrostatique » entre les traits. En plus, un système de règles est destiné à modéliser le fait que l'apparition, la rencontre de certains traits peut en engendrer de nouveaux.

Pour tester la pertinence linguistique et la viabilité de son formalisme, G. Perrier a développé un prototype d'analyseur syntaxique utilisant la programmation par contraintes avec le langage Oz: c'est un langage adapté à ce style de programmation [84], et cela permet de s'inspirer du travail effectué autour des contraintes d'arbres à l'université de Sarrebrück [21]. L'avantage de la programmation par contraintes est de conjuguer efficacité de calcul avec souplesse d'expression. Il est possible par exemple possible de décrire par une seule entrée lexicale toutes les configurations syntaxiques possibles d'un même mot en factorisant les aspects communs et traitant les particularités à l'aide de disjonctions. Le système de contraintes permet ensuite de déduire du contexte de la phrase analysée pour chaque disjonction la composante qui est pertinente plutôt que de le faire a priori.

4.3 Génération et traduction automatique

La génération automatique est une application tout aussi importante que l'analyse et qui pourtant est moins souvent abordée par les formalismes linguistique comparables tels TAGs et HPSG. Il y a une raison à cela: il faut que l'interface syntaxe/sémantique soit aussi automatique que possible, et donc les grammaires catégorielles sont un bon candidat. On notera néanmoins que partir de représentations sémantiques à la Montague est une simplification considérable du problème de la génération ; c'est néanmoins une étape importante, qui, associée à d'autres types de traitement sémantiques plus riches et malléables, permet d'envisager de la *vraie* génération d'énoncés, voire de textes.

Les travaux faits ont utilisé la correspondance entre réseaux de démonstration λ-termes : cela permet d'avoir un même formalisme où décrire les analyses syntaxiques et les représentations sémantiques. La question est la suivante : étant donné un lexique incluant le type syntaxique et la sémantique des mots, si on se donne une représentation sémantique R, comment construire un réseau syntaxique dont la sémantique soit R? Du point de vue logique il s'agit d'inverser le mécanisme d'élimination des coupures (ou β-réduction) à partir de R jusqu'à reconstruire une démonstration dans laquelle on puisse identifier les représentations sémantiques des mots.

Le premier travail en ce sens a été fait par Morrill et Merenciano [50] et utilisait l'unification des λ-termes pour construire l'analyse syntaxique, ce qui menait parfois à tort à des questions indécidables. Au sein du laboratoire de recherche de Xerox, S. Pogodalla a poursuivi ce travail en se basant sur [20], et se propose de réaliser sur cette base un traducteur automatique de phrases comme celles qu'on trouve dans les guides touristiques.

L'architecture de ce programme interactif est la suivante :

1. *L'utilisateur propose une phrase à traduire.*
2. Le système analyse l'énoncé et calcule ses représentations sémantiques. S'il n'en trouve qu'une il exécute l'étape 4 et s'il en trouve plusieurs il produit dans la langue source une phrase non ambiguë pour chaque représentation sémantique.
3. *L'utilisateur choisit la phrase non ambiguë qu'il souhaite traduire.*
4. Le système produit la phrase correspondante dans la langue cible.

Le programme de génération est utilisé tant pour la langue source que pour la langue cible, et suppose que les constantes sémantiques utilisées dans les deux langues se correspondent, ce qui est une forte idéalisation ; néanmoins pour un objectif limité, et si les langues sont assez proches, cela peut fonctionner.

Sa technique utilise une représentation matricielle des réseaux de démonstration où la réduction des démonstrations est un calcul matriciel. Il s'agit d'inverser ce calcul de manière à retrouver des sous-matrices connues (les sémantiques des mots), et il faut tester la correction des réseaux de démonstration obtenus, pour voir s'ils sont vraiment des analyses syntaxiques. Ces algorithmes sont particulièrement agréables car ce sont des algorithmes sur les graphes (pour la correction des réseaux) et sur des matrices (pour inverser la formule d'élimination des coupures). On a ainsi affaire à des structures de données dont l'algorithmique est très développée. Après avoir programmé les algorithmes de vérification de la correction des réseaux en CaML, S. Pogodalla a finalement réalisé ces modules en Objective CaML, qui comporte des aspects objets : c'est un langage plus riche qui permet de retrouver CaML. Pour plus de détails on consultera [72].

5 Liens avec d'autres formalismes

Pour situer les divers formalismes de traitement des langues, on peut, pour simplifier, dire que leur traitement de la syntaxe fait toujours intervenir l'arbre des dépendances et l'ordre linéaire des constituants dans la phrase. Par exemple les grammaires syntagmatiques privilégient l'ordre linéaire (la concaténation des constituants) tandis que les grammaires de dépendance privilégient l'arbre de la hiérarchie des constituants. Ces deux formalismes « extrêmes » ont alors des difficultés à rendre compte de l'ordre qu'elles n'ont pas privilégié. Les grammaires catégorielles tentent de lier ces deux notions, en postulant que la tête d'une expression et son complément sont contiguës (pour la concaténation), ce qui est trop restrictif pour certaines constructions mais, en fait, bien adapté aux plus courantes. C'est à mon avis cette minimalité, et la netteté du formalisme qui expliquent l'influence des grammaires catégorielles sur d'autres formalismes.

En effet, certaines propriétés des grammaires catégorielles ont sciemment été inclues dans d'autres formalismes : le « slash » et plus généralement le traitement de la sous-catégorisation dans GPSG ou HPSG, mais aussi la lexicalisation des

TAGs en LTAGs. Les récentes grammaires de liens [83] rappellent un peu les grammaires catégorielles : planarité des dépendances (liée à la planarité des axiomes qui expriment les dépendances dans les réseaux de démonstration), consommation des traits et connexité. Cependant à ma connaissance ce parallèle n'a pas été précisé, et il nécessiterait de considérer des extensions des grammaires catégorielles telles celles de [33] ou [46] car la contiguïté des constituants est bien plus forte dans le calcul de Lambek originel.

Essentiellement intéressés par l'interface aisée avec la sémantique, et la minimalité de l'approche, d'autres formalismes pour le traitement automatique des langues ont cherché à se rapprocher des grammaires catégorielles. Réciproquement ces dernières ont souhaité capturer des phénomènes linguistiques plus aisés à décrire dans d'autres systèmes et espèrent ainsi se rapprocher d'application réelles comme celles développées avec les TAGs.

5.1 Grammaires de dépendances

La représentation des types sous forme de graphes élémentaires permet de voir un lien assez trivial entre les grammaires catégorielles et les grammaires de dépendances, en particulier les grammaires dites « de mots » [30, 31]. C'est d'ailleurs en partie cette constatation qui a renforcé l'intérêt de l'étude des grammaires catégorielles au travers des réseaux de preuves, puisqu'on peut toujours simplifier les réseaux du calcul de Lambek de manière à confondre les nœuds identifiés par un lien axiome et obtenir finalement de cette façon des structures très proche des graphes de dépendances, comme l'a montré Lecomte [43].

5.2 Grammaires d'Arbres Adjoints (TAG)

Le lien avec les grammaires d'arbres adjoints [1, 35] est essentiellement le fruit de l'étude des réseaux de démonstration pour les grammaires catégorielles (ou de la déduction naturelle, quasi identique dans le cas de calculs intuitionnistes). Ce n'est pas trop étonnant, car les réseaux contiennent des structures d'arbres.

Joshi et Kulick [34] ont décrit les arbres des TAGs comme des preuves partielles dans la déduction naturelle associée au calcul de Lambek. L'opération de substitution correspond au remplacement d'une hypothèse par une preuve partielle, et l'adjonction à l'insertion d'une preuve partielle entre deux nœuds d'un arbre de

déduction partiel. C'est un système où la logique d'assemblage est commutative tandis que celle à l'intérieur des arbres partiels associés aux mots est non commutative : on rejoint les calculs linéaires mixtes [19, 80, 5].

Abrusci, Fouqueré et Vauzeilles [4] ont pour leur part codé les TAGs dans un fragment du calcul de Lambek : cela peut étonner, vu que les grammaires de Lambek ne décrivent que des langages algébriques. De fait, la logique de Lambek est dans ce cas utilisée non pour dériver un type (le type S par exemple) à partir des types associés à des mots comme on le fait dans les grammaires de Lambek, mais pour combiner des fragments d'arbres, où la structure en constituants est déjà préconstruite, ce qui est un problème tout différent. L'adjonction consiste à remplacer une coupure (emplacement pour l'adjonction) par deux coupures, le réseau intermédiaire correspondant à l'arbre qu'on adjoint. La logique linéaire intuitionniste fonctionne ici comme logique *d'assemblage* de structures préexistantes, ces dernières étant formulées dans un autre langage, celui des sous-arbres de dérivation d'une grammaire algébrique. [11]

S. Pogodalla a montré que les grammaires en réseaux de Lecomte et Retoré, sont une extension des LTAGs et lorsqu'on contraint les réseaux apparaissant dans le lexique à avoir une forme simple (qui rappelle celle d'un arbre) elles correspondent exactement aux LTAGs. [71]

5.3 Grammaires Minimalistes

La théorie du gouvernement et du liage de Chomsky qui a permis d'exhiber de nombreux principes linguistiques (critère θ, théorie du Cas, gouvernement des anaphores, etc.) s'est avérée très difficile à formaliser, même si certaines idées se retrouvent dans LFG et HPSG. Il y a cependant eu des tentatives qui se sont appuyées sur des formalismes logiques (voir par exemple [82]). Néanmoins cela restait bien loin des grammaires catégorielles car il s'agissait plus de logique des traits, et de description logique que de systèmes déductifs.

C'est avec le récent programme minimaliste de Chomsky [15, 73] du moins tel qu'il a été formalisé par Stabler [85] que des rapprochements sont apparus, particulièrement prometteurs au vu de la profondeur de la théorie linguistique chom-

11. C'est donc à rapprocher des travaux de Dalrymple et Pereira [17] où des formules de logique du premier ordre sont assemblées par la logique linéaire intuitionniste pour produire les formes sémantiques dans les grammaires LFG [1].

skienne.[12] Il faut dire qu'à la différence de la « théorie du gouvernement et du liage », [73], le minimalisme, même avant toute formalisation présentait des points communs avec les grammaires catégorielles qui sont explicités dans l'introduction de la rencontre *Resource logics and minimalist grammars* [77] :

- Il propose une vue totalement lexicalisée de la grammaire.
- Il s'agit d'une théorie dérivationnelle qui remplace les contraintes sur les représentations intermédiaires par des contraintes d'économie sur la dérivation.
- Les constituants s'assemblent suivant la règle de fusion similaire aux règles de simplification des grammaires AB ou au modus ponens du calcul de Lambek.
- La relation entre « tête » et « non-tête » (spécifieur ou complément) coïncide le plus souvent avec la relation entre fonction et argument dans les grammaires catégorielles.

Reste la notion de déplacement chère à toutes les théories chomskiennes. On a vu au paragraphe 3.1 que les calculs multimodaux à la Moortgat permettent de modéliser dans la logique de tels phénomènes et effectivement, les premières connections ont été établies dans ce cadre : [16, 45, 29, 92]. Néanmoins, on notera que la variation langagière est expliquée chez Chomsky par des *paramètres*, sortes de variables booléennes associées à des traits, là où les grammaires catégorielles utilisent plus volontiers des postulats : ces solutions fonctionnent toutes les deux, mais sont elles conciliables ?

En restant plus proches de la vision chomskienne, et en s'autorisant un mécanisme de calcul extra logique (un automate) Lecomte et Retoré [48] ont modélisé les grammaires minimalistes dans le calcul de Lambek originel. Cela peut surprendre car les grammaires minimalistes ont le même pouvoir d'expression que les MC-TAGS [51] et engendrent des langages non algébriques, tandis que les grammaires de Lambek correspondent aux langages algébriques. L'idée est de représenter dans le calcul de Lambek les dépendances, et de calculer l'ordre des mots par l'automate : celui-ci parcourt et étiquette la démonstration par des chaînes de mots. Le mouvement est capturé par le lien qui unit les deux hypothèses déchargées lors de la règle d'élimination du produit. L'automate, lorsqu'il traite cette règle, fait la distinction

12. En tout cas d'un point de vue cognitif et psycholinguistique cette théorie est assurément la plus développée, puisque c'est pour ainsi dire la seule et qu'elle bénéfi cie de quarante ans de recherches intenses.

entre traits forts et faibles : les catégories correspondant aux traits faibles induisent un mouvement « furtif » (des traits sémantiques) tandis que celles associées aux traits forts induisent un mouvement « visible » (des traits sémantiques et phonologiques), tout comme dans [15, 73, 85]. Pour prendre un exemple, la différence entre une langue SVO et une langue SOV est alors modélisée lexicalement par le cas fort ou faible que donne le verbe à son objet, c'est-à-dire par un type verbe comportant le type atomique *CASE* ou *case* cette différence étant prise en compte lors de la lecture de la démonstration par l'automate.

6 Conclusion

Dresser un bilan des grammaires catégorielles serait un peu prématuré, aussi je préfère indiquer quelques directions actuelles de recherches qui, personnellement, me paraissent importantes ou me plaisent :

- Il semble essentiel d'intégrer ce formalisme jusqu'ici assez confidentiel à d'autres plus répandus et impliqués dans des applications réelles : les TAGs semblent le formalisme le plus proche, mais encore faudrait-il savoir réutiliser les lexiques TAGs automatiquement pour constituer un lexique catégoriel d'une taille raisonnable.
- Dans le même ordre d'idées, il semble aujourd'hui essentiel de programmer des modules d'analyse et de génération qui utilisent ce formalisme. Pour que cela soit concluant il faut choisir des conditions où il est plus adapté que d'autres, c'est-à-dire des circonstances où le temps de calcul importe moins que la sûreté et la précision, et où profiter de l'interface avec la sémantique de Montague (si possible associée à un traitement d'autres types d'informations sémantiques). La génération et l'aide à la traduction (qui est très liée) sont de bons exemples de traitement des langues où l'on ne peut se contenter d'analyses syntaxique superficielles.
- A un niveau plus théorique, le lien avec le programme minimaliste est à renforcer ; la description de nombreux phénomènes linguistiques en découlera (pour les grammaires catégorielles) et la formalisation logique permettra de tester les assertions calculatoires faites par cette théorie linguistique.
- Dans la description de constructions linguistiques, les pronoms clitiques du français me semblent un défi pour les formalismes grammaticaux. Certes les

grammaires multimodales ont proposé une solution déjà mentionnée, mais on aimerait un traitement plus léger et d'un temps de calcul plus réaliste, par exemple dans le cadre des grammaires en réseaux. Y arriver, par exemple en s'inspirant de travaux chomskyens, serait un réel bénéfice pour les grammaires catégorielles, car à notre connaissance il n'y a pas de modélisation élégante de ces questions dans aucun formalisme de traitement automatique des langues.

– L'apprentissage est un domaine trop souvent ignoré par les formalismes grammaticaux, et pour lequel les grammaires catégorielles sont particulièrement adaptées. Les algorithmes existent pour les grammaires AB, sont implantés, et on souhaite les tester sur des exemples réels. L'étape suivante sera d'étendre ces algorithmes aux grammaires de Lambek et de tirer pleinement profit de la correspondance entre syntaxe et sémantique pour avoir (peut-être) un modèle réaliste de l'apprentissage par l'enfant de sa langue maternelle : un joli sujet tant sur le plan formel que cognitif.

Espérons que ce panorama et ces quelques questions donneront envie à certains lecteurs de consulter les ouvrages, ou articles plus conséquents cités, et de contribuer eux-mêmes au développement de ce domaine.

REMERCIEMENTS : Les commentaires de M. Abrusci, M. Dymetman, Y. Le Nir, M. Moortgat, R. Moot, G. Morrill, J. Nicolas, G. Perrier, S. Pogodalla, P. Sébillot et tout particulièrement ceux d'Alain Lecomte m'ont considérablement aidé à réaliser ce panorama.

Références

[1] Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, 1993.

[2] Michele Abrusci and Claudia Casadio, editors. *Third Roma Workshop: Proofs and Linguistics Categories – Applications of Logic to the analysis and implementation of Natural Language*. Bologna:CLUEB, 1996.

[3] V. Michele Abrusci. Phase semantics and sequent calculus for pure noncommutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.

[4] V. Michele Abrusci, Christophe Fouqueré, and Jacqueline Vauzeilles. Tree adjoining grammar and non-commutative linear logic. In Retoré [75], pages 13–17.

[5] V. Michele Abrusci and Paul Ruet. Non-commutative logic i: the multiplicative fragment. *Annals of Pure and Applied Logic*, 101(1):29–64, 1999.

[6] Pieter Adriaans. Grammar induction as substructural inductive logic programming. In *Learning Language in Logic workshop*, pages 117–127, Bled, June 1999. http://www.cs.york.ac.uk/mlg/lll/workshop/.

[7] Kasimir Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935.

[8] Y. Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[9] Y. Bar-Hillel, C. Gaifman, and E. Shamir. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel*, F(9):1–16, 1963.

[10] Philippe Blache. Spécifications de contraintes syntaxiques par filtrage de réseaux de preuve. Rapport de recherche, Université d'Aix-en-Provence, 1999.

[11] Patrick Blackburn, Marc Dymetman, Alain Lecomte, Aarne Ranta, Christian Retoré, and Eric Villemonte de la Clergerie. Logical aspects of computational linguistics: an introduction. In Retoré [75], pages 1–20.

[12] W. Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [90], chapter 12, pages 683–736.

[13] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[14] Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

[15] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[16] Thomas Cornell. Derivational and representational views of minimalist transformational grammar. In Alain Lecomte, François Lamarche, and Guy Perrier, editors, *Logical aspects of computational linguistics: Second International Conference, LACL '97, Nancy, France, September 22–24, 1997; selected papers*, volume 1582. Springer-Verlag, 1999.

[17] M. Dalrymple, J. Lamping, F. Pereira, and V. Saraswat. Linear logic for meaning assembly. In Morrill and Oehrle [60], pages 75–93.

[18] P. De Groote. An algebraic correctness criterion for intuitionistic proof-nets. In *Logical Fundations of Computer Science, LFCS'97*, volume 1234 of *Lecture Notes in Computer Science*, pages 130–140, 1997.

[19] Philippe de Groote. Partially commutative linear logic: sequent calculus and phase semantics. In Abrusci and Casadio [2], pages 199–208.

[20] Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn Morrill, and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, August 1996. FoLLI.

[21] D. Duchier and Gardent C. A constraint-based treatment of descriptions. In *ICOS'99 (Inference in Computational Semantics)*, Amsterdam, 1999.

[22] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[23] Jean-Yves Girard. Linear logic: its syntax and semantics. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*, pages 1–42. Cambridge University Press, 1995.

[24] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[25] L.R. Gleitman and M. Liberman, editors. *An invitation to cognitive sciences, Vol. 1: Language*. MIT Press, 1995.

[26] L.R. Gleitman and E.L. Newport. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman and Liberman [25], chapter 1, pages 1–24.

[27] E.M. Gold. Language identification in the limit. *Information and control*, 10:447–474, 1967.

[28] M. Hepple. *The Grammar and Processing of Order and Dependency, a categorial approach*. Phd thesis, Centre of Cognitive Science, Edinburgh, 1990.

[29] Dirk Heylen. *Types and Sorts – Resource Logic for Feature Checkimg*. PhD thesis, Universiteit Utrecht, 1999.

[30] R. Hudson. *Word Grammar*. Blackwell, Oxford, 1984.

[31] R. Hudson. *English Word Grammar*. Blackwell, Oxford, 1990.

[32] Mark E. Johnson. Proof nets and the complexity of processing center-embedded constructions. *Journal of Logic Language and Information*, 7(4):433–447, 1998.

[33] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. In Morrill and Oehrle [60], pages 138–149.

[34] Aravind Joshi and Seth Kulick. Partial proof trees, resource sensitive logics and syntactic constraints. In Retoré [75], pages 21–42.

[35] Aravind Joshi and Yves Schabes. Tree adjoining grammars. In Rozenberg and Salomaa [79], chapter 2.

[36] Aravind Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Schieber, and T. Wasow, editors, *Fundational issues in natural language processing*. MIT Press, 1991.

[37] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

[38] Esther Kraak. A deductive account of french objects clitics. *SYntax and Semantics*, 30:271–312, 1998.

[39] Natasha Kurtonina and Michael Moortgat. Structural control. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures*, pages 75–113. CSLI, 1997. Distributed by Cambridge University Press.

[40] François Lamarche and Christian Retoré. Proof nets for the lambek calculus – an overview. In Abrusci and Casadio [2], pages 241–262.

[41] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, 65:154–169, 1958.

[42] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[43] Alain Lecomte. Proof-nets and dependencies. In *COLING*, pages 394–401. Nantes, August 1992.

[44] Alain Lecomte. Grammaire et théorie de la preuve: une introduction. *Traitement Automatique des Langues*, 37(2):1–38, 1996.

[45] Alain Lecomte. Proof-nets, hybrid logics and minimalist representations. In *Mathematics of Language 6*, 1999.

[46] Alain Lecomte and Christian Retoré. Pomset logic as an alternative categorial grammar. In Morrill and Oehrle [60], pages 181–196.

[47] Alain Lecomte and Christian Retoré. Words as modules: a lexicalised grammar in the framework of linear logic proof nets. In Carlos Martin-Vide, editor, *Mathematical and Computational Analysis of Natural Language — selected papers from ICML'96*, volume 45 of *Studies in Functional and Structural Linguistics*, pages 129–144. John Benjamins publishing company, 1998.

[48] Alain Lecomte and Christian Retoré. Towards a minimal logic for minimalist grammars: Another use of lambek calculus. In *Formal Grammar, FG'99*, pages 83–92. FoLLI, 1999.

[49] Marie-Ange Légeret. *Algèbres de démonstrations en grammaires catégorielles*. Thèse de Doctorat, spécialité Mathématiques, Université Blaise Pascal, Clermont-Ferrand, janvier 1996.

[50] Josep Maria Merenciano and Glyn Morrill. Generation as deduction. In Retoré [75], pages 77–80.

[51] Jens Michaelis. Derivational minimalism is mildly context-sensitive. In Moortgat [55].

[52] Richard Montague. *The collected papers of Richard Montague, edited and with an introduction by Richmond Thomason*. Yale University Press, 1974.

[53] Michael Moortgat. *Categorial Investigations*. Foris, Dordrecht, 1988.

[54] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [90], chapter 2, pages 93–177.

[55] Michael Moortgat, editor. *Logical Aspects of Computational Linguistics, LACL'98, selected papers*, LNCS/LNAI. Springer-Verlag, 1999.

[56] Richard Moot. Grail: An automated proof assistant for categorial grammar logics. In R.C. Backhouse, editor, *Proceedings of the 1998 User Interfaces for Theorem Provers Conference*, pages 120–129, 1998.

[57] Richard Moot. User's guide to Grail 2.0. Manuscript, 1998.

[58] Richard Moot and Quintijn Puite. Proof-nets for the multimodal lambek calculus. In G.-J. Kruiff and Richard T. Oehrle, editors, *Formal Grammar '99*. FoLLI, 1999. http://www.math.uu.nl/publications/preprints/1096.ps.gz.

[59] Glyn Morrill. Incremental processing and acceptability. Research Report LSI-98-48-R, Universitat Politècnica di Catalunya, 1998.

[60] Glyn Morrill and Richard Oehrle, editors. *Formal Grammar*, Barcelona, August 1995. FoLLI.

[61] Glyn V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

[62] Jacques Nicolas. Grammatical inference as unification. Rapport de Recherche RR-3632, INRIA, 1999. http://www.inria.fr/RRRT/publications-eng.html.

[63] Barbara Partee. Montague grammar. In van Benthem and ter Meulen [90], chapter 1, pages 5–92.

[64] Mati Pentus. Lambek grammars are context-free. In *Logic in Computer Science*. IEEE Computer Society Press, 1993.

[65] Mati Pentus. Product-free lambek calculus and context-free grammars. *Journal of Symbolic Logic*, 62(2):648–660, 1997.

[66] G. Perrier. Labelled proof nets for the syntax and semantics of natural languages. *Journal of the IGPL*, 7(5):629–654, 1999.

[67] Guy Perrier. From intuitionistic proof nets to interaction grammars. Technical Report 99-R-120, LORIA, Nancy, France, 1999.

[68] Guy Perrier. Labelled proof nets for the syntax and semantics of natural languages. *Logic Journal of the IGPL*, 7(5):629–654, September 1999.

[69] Steven Pinker. *The Language Instinct*. Penguin Science, 1994.

[70] Steven Pinker. Language acquisition. In Gleitman and Liberman [25], chapter 6, pages 135–182.

[71] Sylvain Pogodalla. Ltags and pomset logic. In Moortgat [55].

[72] Sylvain Pogodalla. Generation with semantic proof-nets. Rapport de recherche, INRIA, janvier 2000. http://www.inria.fr/RRRT/RR-3878.html.

[73] Jean-Yves Pollock. *Langage et cognition: le programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.

[74] Christian Retoré. Calcul de lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

[75] Christian Retoré, editor. *Logical Aspects of Computational Linguistics, LACL'96*, volume 1328 of *LNCS/LNAI*. Springer-Verlag, 1997.

[76] Christian Retoré. Pomset logic: a non-commutative extension of classical linear logic. In Ph. de Groote and J. R. Hindley, editors, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318, 1997.

[77] Christian Retoré and Edward Stabler. Resource logics and minimalist grammars: introduction. In Christian Retoré and Edward Stabler, editors, *Resource Logics and Minimalist Grammars*, European Summer School in Logic Language and Information, Utrecht, 1999. FoLLI. RR-3780 http://www.inria.fr/RRRT/publications-eng.html.

[78] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

[79] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer Verlag, Berlin, 1997.

[80] Paul Ruet. *Logique non-commutative et programmation concurrente*. Thèse de doctorat, spécialité logique et fondements de l'informatique, Université Paris 7, 1997.

[81] Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45, 1997.

[82] Pascale Sébillot. *Modélisation de filtrage par arbres et par traits dans les grammaires logiques pour l'analyse automatique du langage naturel*. Thèse de doctora, spécialité informatique, Institut National des Sciences Appliquées, Rennes, 1989.

[83] D. D. Sleator and D. Temperley. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*, page ??, 1993. Available through: ftp://bobo.link.cs.cmu.edu/pub/sleator/link-grammar/.

[84] Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.

[85] Edward Stabler. Derivational minimalism. In Retoré [75], pages 68–95.

[86] Mark Steedman. Combinators and grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*. Reidel, Dordrecht, 1988.

[87] Isabelle Tellier. Meaning helps learning syntax. In *Fourth International Colloquium on Grammatical Inference, ICG'98*, 1998.

[88] Hans-Jörg Tiede. Lambek calculus proofs and tree automata. In Moortgat [55].

[89] Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI, 1992. (distributed by Cambridge University Press).

[90] J. van Benthem and A. ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

[91] Johan van Benthem. *Language in Action: Categories, Lambdas and Dynamic Logic*, volume 130 of *Sudies in logic and the foundation of mathematics*. North-Holland, Amsterdam, 1991.

[92] Willemijn Vermaat. *Controlling Movement: Minimalism in a Deductive Perspective*. Doctorandus thesis, Universiteit Utrecht, 1999.

[93] W. Zielonka. Axiomatizability of the Adjukiewicz-Lambek calculus by means of cancellation schemes. *Zeitschrift für matematische Logik und Grundlagen der Mathematik*, 28:539–548, 1982.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399