Verbum XL, 2018, n° 2, 237-267

L'APPROCHE LOGIQUE DES GRAMMAIRES CATÉGORIELLES : UNE SYNTAXE TOURNÉE VERS LA SÉMANTIQUE [1]

Christian RETORÉ
LIRMM, Université de Montpellier, CNRS

Cet article est dédiée à la mémoire d'Isabelle Tellier (1968-2018), amie avec laquelle j'eus la chance de travailler sur l'apprentissage des grammaires catégorielles (voir par exemple Bechet *et al.* 2007).

RÉSUMÉ

L'approche logique des grammaires catégorielles renouvelle les travaux des années cinquante (voire trente) et rapproche les formalismes catégoriels des systèmes de typage utilisés en informatique. La linguistique informatique s'est ainsi enrichie d'une approche originale qui néanmoins rejoint d'autres formalisations plus connues, telles les grammaires d'arbres adjoints, les grammaires de dépendances et certaines grammaires transformationnelles. 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 ces formalismes et le lien privilégié avec la sémantique qui en résulte. Les résultats de base sont explicitement traités tandis que les recherches et réalisations récentes ne sont qu'esquissées.

ABSTRACT

The logical approach to categorial grammars renews ideas that go back to the fifties (and even to the thirties) and recent research connects categorial formalisms to 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 and the tight connection with semantics. Basic results are precisely covered while more recent research and insights are only informally presented.

---

[1] Cet article reprend, avec l'autorisation de la revue *Technique et Science Informatiques*, certains passages de Retoré (2001).

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. À ces deux aspects correspondent deux familles de méthodes, qui restent difficilement conciliables 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 (Pollock 1997).

Inscrite dans cette approche calculatoire et symbolique, notre présentation concerne surtout les grammaires catégorielles apparues il y a bien longtemps (Ajdukiewicz 1935, Bar-Hillel 1953), ainsi que leurs extensions sous la forme générale de ce qu'il est convenu d'appeler les « grammaires de types logiques » (Morrill 1994, Moortgat 1997). C'est un modèle de la syntaxe des langues, mais qui entretient d'étroits rapports avec la sémantique de Montague (Montague 1964, Partee 1997). 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.

Les grammaires catégorielles entrent dans le cadre d'un lexicalisme radical ; comme dans les grammaires d'arbres adjoints lexicalisées (Joshi 1997, Abeillé 1993), les règles sont uniformes, données une fois pour toutes et indépendantes de la langue, une sorte de grammaire universelle à la Chomsky (Chomsky 1995, Pollock 1997). Seule la variation des catégories que le lexique associe à des entrées lexicales – ou les postulats globaux associés à certains opérateurs dans les calculs multimodaux de Moortgat (1997) –, exprime les variations d'une langue à l'autre. On notera que les travaux récents de Chomsky (Chomsky 1995, Pollock 1997) 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 apporte élégance et simplicité conceptuelle à la grammaire, mais ce n'est pas son principal avantage. En effet, cette division de la grammaire en un lexique et quelques règles universelles est d'un immense 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. D'un point de vue théorique, c'est un des piliers des théories chomskyennes (Pollock 1997, Pinker 1995), ce 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 (Gold 1967), mais en plein essor depuis la fin des années 1980, que ce soit pour la génomique, le diagnostic ... ou le traitement des langues. Les grammaires lexicalisées, et en particulier les grammaires catégorielles, 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. Dans le cas des grammaires catégorielles déductives, une présentation assez récente se trouve dans Bonato et Retoré (2014) et un panorama plus ancien, incluant l'apprentissage d'autres formalismes grammaticaux en rapport avec les Grammaires catégorielles, se trouve dans Bechet *et al*. (2007).

Techniquement, les grammaires catégorielles, depuis leur formulation logique par Lambek (1958), et plus encore dans les extensions récentes, s'apparentent plus à la théorie de la démonstration (Girard 1988) qu'à la théorie des langages classiques (Rozenberg et Salomaa 1997), ou qu'aux autres formalismes de traitement automatique des langues (Abeillé 1993) ; cela explique en partie leur relatif isolement. L'autre raison de la discrétion des grammaires catégorielles déductives jusqu'aux années 80 est que le calcul de Lambek est syntaxiquement trop restrictif, et qu'on ne connaissait pas de liens entre ce système déductif (linéaire avant l'heure) et les logiques usuelles, classique, intuitionniste ou modale. Les extensions des grammaires de Lambek, syntaxiquement plus riches, sont apparues en reliant le calcul de Lambek original à d'autres logiques comme la logique modale (Van Benthem 1991, Moortgat 1997), puis à la logique linéaire (Retoré 1996) inventée par Girard (1987) ; ces connexions sont reprises dans Moot et Retoré (2012).

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 (1953) dans le style des calculs de combinateurs, Steedman (1988) propose une toute autre approche des grammaires catégorielles qui étend leur capacité générative aux langages légèrement contextuels (*mildly context*

sensitive) (Joshi 1991). Celles-ci sont présentées dans un autre article du même numéro.

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 volume *Grammaire et théorie de la preuve* (Lecomte 1996), le *Handbook of Logic and Language* (Van Benthem et ter Meulen 1997, 2010), ou les conférences *Logical Aspects of Computational Linguistics* (Blackburn 1997, Retoré 2016). Ces références élargissent le lien entre logique, langues et langages, notamment grâce à l'interface limpide entre syntaxe catégorielle et sémantique compositionnelle et à la notion de déduction logique.

L'article que nous proposons est organisé ainsi :

– la section 2 part des grammaires AB, pour présenter le calcul de Lambek, le premier système déductif à modéliser la grammaire d'une langue ;

– la section 3 expose l'algorithme de calcul d'une représentation sémantique à partir d'une analyse syntaxique dans le calcul de Lambek ;

– la section 4 discute des extensions du calcul de Lambek qui augmentent son pouvoir d'expression syntaxique tout en préservant la correspondance avec la sémantique ;

– la section 5 présente une de ces extensions, les grammaires catégorielles multimodales, ainsi que la plateforme Grail d'analyse syntaxique et sémantique du français à large échelle ;

– la section 6 présente une autre extension dont l'intérêt, plutôt théorique, est de se rapprocher du programme minimaliste de la grammaire générative.

2. DES GRAMMAIRES CATÉGORIELLES CLASSIQUES AU CALCUL DE LAMBEK

2.1. Les grammaires AB

Nous reprenons ici les grammaires AB qui sont décrites dans l'introduction de ce numéro, mais sous un angle qui ouvre la porte aux grammaires de type logique ou déductives. Nous suivons le livre de Moot et Retoré (2012) qui contient bien plus de détails et d'exemples. Les premières grammaires catégorielles, souvent appelées grammaires AB, nommées ainsi après les travaux d'Ajdukiewicz (1935) et de Bar-Hillel (1953), procèdent comme suit. On se donne un ensemble fini P de catégories de base, par exemple sn, n, S (respectivement pour syntagme nominal, nom, phrase) dont l'un joue un rôle particulier : S (l'analogue du symbole non-terminal S dans une grammaire syntagmatique). Les catégories syntaxiques sont obtenues à partir des catégories de base par application de deux opérateurs / et \. Une catégorie complexe comme (sn\S)/sn est celle d'une expression attendant un syntagme

nominal à droite, puis un syntagme nominal à sa gauche pour constituer une phrase, c'est la catégorie d'un verbe transitif[2].

Un petit nombre de règles universelles (indépendantes de la grammaire de la langue décrite) de calcul sur les catégories, correspond à un calcul non commutatif de fractions. Nous allons ici reformuler les règles des grammaires AB (présentées dans le premier article de ce numéro), avec des expressions appelées séquents.

Un séquent $C_1,\ldots,C_n \vdash X$ est composé d'une suite de catégories $C_1,\ldots,C_n$ à gauche du signe $\vdash$ (qui se lit « entraîne ») et d'une catégorie X à droite de $\vdash$. Une séquent se comprend ainsi : si n expressions (généralement des mots) ont pour catégories respectives $C_1,\ldots,C_n$, alors la suite de ces n expressions, dans son intégralité, est de catégorie X. On dit aussi que la suite de catégories $C_1,\ldots,C_n$ entraîne la catégorie X.

Avant tout commentaire, donnons immédiatement les règles des grammaires AB de cette manière, en déduction naturelle. L'axiome est e point de départ de toute dérivation :

$$A \vdash A$$

Si une expression est de catégorie A, alors elle est de catégorie A. Difficile de ne pas approuver ! Les deux premières règles sont tout simplement celles des grammaires AB, mais formulées avec des suites de catégories :

La règle d'élimination de \ , désignée par \e :

$$\frac{G_1,\ldots,G_n \vdash A \qquad D_1,\ldots,D_p \vdash A\backslash B}{G_1,\ldots,G_n, D_1,\ldots,D_p \vdash B}\ \backslash e$$

Si une suite de catégories $G_1,\ldots,G_n$ entraîne la catégorie A, et si une suite de catégories $D_1,\ldots,D_p$ entraîne la catégorie A\B, alors la suite de catégories $\Gamma = G_1,\ldots,G_n$, et $\Delta = D_1,\ldots,D_p$ entraine la catégorie B. On reconnait la règle 'A (A\B) → B' des grammaires AB, reformulée avec des suites de catégories.

La règle d'élimination de / désignée par /e :

$$\frac{D_1,\ldots,D_p \vdash B/A \qquad G_1,\ldots,G_n \vdash A}{D_1,\ldots,D_p, G_1,\ldots,G_n \vdash B}\ /e$$

---

2 Prenons garde aux notations : dans les grammaires catégorielles combinatoires (GCC) de Steedman (1988), 'A\B' est noté 'B\A' de sorte que la catégorie correspondant au but est toujours positionnée en première position ; le schéma de réduction se formule alors :
A (B\A) ⊢ B.
Les notations utilisées dans cet article ne sont pas celles de Steedman, mais celle de Bar-Hillel et Lambek.

Cette deuxième règle des grammaires AB pour / , se paraphrase comme nous l'avons fait ci-dessus pour la règle \e.

On retrouve aisément chacune des deux règles des grammaires AB telles qu'elles sont formulées dans le premier article de la revue, à partir d'une dérivation constituée de deux axiomes et de la règle correspondante sur les séquents :

$$\frac{B/A \vdash B/A \qquad A \vdash A}{B/A,\ A \vdash B}\ /e \qquad \frac{A \vdash A \qquad A\backslash B \vdash A\backslash B}{A,\ A\backslash B \vdash B}\ \backslash e$$

Un lexique associe à chaque mot une ou plusieurs catégories. 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 une catégorie C_i, extraite de l'ensemble des catégories, que le lexique attribue à m_i, et tel que la séquence $C_1,\ldots,C_n \vdash S$ soit dérivable à partir des axiomes $X \vdash X$ et des deux règles ci-dessus.

Donnons un exemple minuscule de lexique / grammaire (en italien car l'absence manifestée du sujet permet de trouver un exemple plus réduit qu'en français), afin de manipuler les grammaires AB avec des séquents :

Mot	**Catégorie(s)**
guarda	S/Sinf
passare	Sinf/sn
il	sn/n
treno	n
cosa	S/(S/sn)

La phrase *guarda passare il treno* (« il.elle regarde passer le train ») appartient au langage engendré car :

S/ Sinf, Sinf /sn, sn/n, n ⊢ S

est dérivable de la façon suivante :

$$\dfrac{S/Sinf \vdash S/Sinf \qquad \dfrac{Sinf/sn \vdash Sinf/sn \qquad \dfrac{sn/n \vdash sn/n \quad n \vdash n}{sn/n,\ n \vdash sn}\ /e}{Sinf/sn,\ sn/n\ ,\ n \vdash Sinf}\ /e}{S/Sinf,\ Sinf/sn,\ sn/n,\ n \vdash S}\ /e$$

La phrase *cosa guarda passare* (« que regarde-t-il/elle passer ? ») n'appartient pas au langage engendré, puisqu'on ne peut pas dériver :

S/(S/sn), S/Sinf, Sinf/sn ⊢* S

du fait que S/(S/sn), Sinf/sn ⊢* S/sn n'est pas dérivable, sans quoi *cosa guarda passare* serait reconnue. Certaines règles supplémentaires seraient par conséquent, les bienvenues ; par exemple :

– la montée de type, largement présente dans les grammaires de Montague (*cf.* ci-après) : sn ⊢ (S/(sn\S)) ;

– la composition, (a / b) / (b / c) ⊢ (a /c), ce qui aurait permis de reconnaître la phrase *cosa guarda passare.*

Jusqu'à quelles limites convient-il d'ajouter des règles ? Il faut veiller à ne pas aller jusqu'à introduire une incohérence dans l'ensemble de ces règles, par exemple, une incohérence qui se traduirait par l'identification des deux symboles orientés / et \ ; voir, par exemple, la discussion dans Moortgat (1988).

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

En 1958, Joachim Lambek (1958) montre qu'on peut formaliser les grammaires catégorielles sous une forme logique (peut-être préfèrerait-il dire « catégorique »), dans un système qui permet de dériver toutes les règles de calcul envisagées ci-dessus. Par exemple, la règle de simplification des fractions est alors vue comme la règle du *modus ponens* : A, A⇒B ⊢ B. Le fonctionnement d'une grammaire de Lambek est le même que celui des grammaires AB :

– comme précédemment le lexique associe à chaque mot m_i, une ou plusieurs catégories, celles-ci étant celles des grammaires AB définies précédemment ;
– 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 une catégorie t_i parmi celles que le lexique attribue au mot m_i, telle que la séquence $t_1,\ldots,t_n$ se réduise en S, grâce aux règles du calcul de Lambek.

Les deux nouvelles règles introduites par Lambek sont :

La règle d'introduction de \ , désignée par \i[3] :

---

[3] Il y a en fait une restriction sur les règles /i et \i : il faut que $p \geq 1$, c'est-à-dire qu'il reste des hypothèses, après application de la règle : il ne faut pas attribuer de catégorie à une suite vide de mots. Cette restriction imposée sur les règles /i et \i suffit à assurer qu'aucun séquent sans hypothèse (de la forme ⊢ X) n'est dérivable. Appelons momentanément L1 le calcul de Lambek sans cette restriction, pour voir que cette restriction est nécessaire. La signification de $D_1,\ldots,D_p$ ⊢ A\B est : la suite de catégories $D_1,\ldots,D_p$ doit nécessairement être précédée d'une suite de catégories $G_1,\ldots,G_n$ de catégorie A à sa gauche pour donner

$$\frac{A, D_1,...,D_p \vdash B}{D_1,...,D_p \vdash A\backslash B}\ \backslash i$$

Cette règle se paraphrase ainsi : si la suite de catégories A, $D_1,...,D_p$ entraîne B, alors la suite de catégories $D_1,...,D_p$ entraîne A\B. On comprend bien que cette règle vient compléter la règle \e des grammaires AB : les grammaires AB disent que A suivi de $D_1,...,D_p$ qui entraine A\B est de catégorie B. La règle de Lambek dit que si A suivi de $D_1,...,D_p$ est de catégorie B, c'est que $D_1,...,D_p$ entraîne la catégorie A\B.

La règle d'introduction de / appelée /i :

$$\frac{D_1,....,D_p, A \vdash B}{D_1,...,D_p \vdash B/A}\ /i$$

Cette règle se paraphrase de la même manière que la précédente. Elle répond à la règle /e des grammaires AB.

Voyons maintenant comment, avec le lexique précédemment donné, il devient possible d'analyser, dans le calcul de Lambek, *cosa guarda passare* (voir la Figure 1).

$$\dfrac{S/(S/sn) \vdash S/(S/sn) \qquad \dfrac{\dfrac{S/Sinf \vdash S/Sinf \qquad \dfrac{Sinf/sn \vdash Sinf/sn \qquad sn \vdash sn}{Sinf/sn, sn \vdash Sinf}\ /e}{S/Sinf, Sinf/sn, sn \vdash S}\ /i}{S/Sinf, Sinf/sn \vdash S/sn}\ /e}{S/(S/sn),\ S/Sinf,\ Sinf/sn \vdash S}\ /e$$

Figure 1. – Analyse de *« cosa guarda passare »* dans le calcul de Lambek

Cet exemple, repose sur la composition de deux expressions utilisant le symbole / : S/Sinf, Sinf/sn ⊢ S/sn est dérivable dans le calcul de Lambek grâce à la règle d'introduction /i, comme on le voit dans la partie grisée de la dérivation. L'étape importante est la règle /*i* et les règles préalables situées au-dessus dans l'arbre. On suppose que *guarda passar* est suivi d'un 'sn' inconnu '*X*' (*Margarita, il treno,...*) et on obtient une phrase (de catégorie S)

---

une chaîne $D_1,...,D_p$, $G_1,...,G_n$ de catégorie B. Si dans L1 on peut dériver ⊢ A alors de $D_1,...,D_p \vdash A\backslash B$ on peut déduire $D_1,...,D_p \vdash A\backslash B$, c'est-à-dire que la suite $G_1,...,G_n$ fournie à gauche de $D_1,...,D_p$ est la suite vide ! L'exemple linguistique suivant montre qu'on rencontre dans la langue des cas où cette restriction est nécessaire ; le lexique pose *très : (n\n)/(n\n)*. Cela permettrait, en l'absence de la restriction, de montrer que *« une femme très »* est un syntagme nominal ; en effet la catégorie « adjectif à droite » (n\n) (par ex. *intelligente*) demandée à la droite de *très* est dérivable (X\X) et ainsi l'adjectif à droite pourrait être, sans la restriction, « rien », la suite vide de mot(s).

'*guarda passare X*' en utilisant la règle /e des grammaires AB. C'est là où on utilise la règle d'introduction /*i* spécifique au calcul de Lambek : si '*guarda passare X'*, avec '*X'* de catégorie 'sn', est de catégorie S, c'est que '*guarda passare'* est de catégorie 'Sinf/sn'. L'attribution de la catégorie 'Sinf/sn' à '*guarda passare*' est établie en introduisant un 'sn' fictif après *passare*, qui est ensuite abstrait par une règle d'introduction : ce 'sn' fictif, positionné après '*passare*', ressemble à la trace d'un constituant déplacé dans les théories chomskyennes : *« Cosa$_i$ guarda passare t$_i$ » (« Que$_i$ regarde-t-il passer t$_i$ »),* alors qu'il n'y a pas ni trace(s), ni déplacement(s) dans le calcul de Lambek.

2.3. Propriétés du calcul de Lambek et des grammaires de Lambek

En tant que système déductif de logique formelle dont les propositions sont les catégories, le calcul de Lambek est :

– *Intuitionniste.* Cette logique est intuitionniste, chaque séquent comporte au plus une formule / catégorie à droite – intuitivement cela signifie que ce calcul ne dérive pas de jugement disjonctif (« cette expression est de catégorie A ou B »). Cela n'empêche pas qu'une expression puisse avoir plusieurs catégories, les mots eux-mêmes peuvent avoir plusieurs catégories.

– *Non commutatif.* Une innovation, du point de vue logique en tout cas : les propositions / catégories ne permutent pas entre elles : '*la pomme*' est un syntagme nominal mais '*pomme la*' n'en est pas un ('sn/n, n ⊢ sn' est dérivable, mais 'n, sn/n ⊢* sn' n'est pas dérivable). En fait, la propriété de linéarité qui suit est nécessaire pour avoir une logique non commutative avec les propriétés d'une logique, mais il est difficile d'expliquer cela en quelques phrases.

– *Linéaire.* Ce calcul ne comporte aucune des règles dites structurelles de la logique : deux catégories / propositions ne peuvent être considérées comme une seule, et on ne peut ajouter de catégorie / proposition. La phrase *'Pierre regarde Marie*' est une phrase car 'sn, ((sn\S)/sn), sn ⊢ S' est dérivable. En revanche, '*Pierre regarde*' n'est pas une phrase car 'sn, ((sn\S)/sn) ⊢* S' n'est pas dérivable puisqu'il manque un 'sn' ; dans une logique usuelle, ce 'sn' pourrait être ajouté puis déplacé et fusionné avec le 'sn' associé à *Pierre* et la suite de catégories deviendrait dérivable. De même, *'Pierre regarde Marie Jacques'* n'est pas une phrase car 'sn, ((sn\S)/sn), sn, sn ⊢* S' n'est dérivable car le 'sn' de Jacques est en trop, alors que dans une logique usuelle, il pourrait être ajouté après la dérivation 'sn, ((sn\S)/sn), sn ⊢ S', qui correspond à l'analyse de *'Pierre regarde Marie'* puis contracté avec le dernier 'sn'.

Les propriétés du calcul de Lambek et des grammaires de Lambek sont exposées scolairement dans les trois premiers chapitres de Moot et Retoré (2012), et s'il n'en faut retenir que certaines, voici les plus importantes :

– *Langages engendrés.* Les linguistes s'accordent pour dire que les langages formels correspondant aux langues humaines sont les langages légèrement contextuels, c'est-à-dire qu'ils nécessitent des formalismes allant un peu au-delà des langages hors-contexte (Joshi *et al*. 1991). Malheureusement, le langage engendré par une grammaire de Lambek est toujours un langage hors-contexte (on dit aussi un langage algébrique) comme l'a montré Pentus (1993). Alors, est-ce beaucoup de bruit pour rien ? Non, car les langages d'arbres, les dérivations arborescentes engendrées par les grammaires de Lambek, comme dans la dérivation en Figure 1 pour *« cosa guarda passare* », sont plus riches que ceux engendrés par les grammaires hors-contexte. Comme l'a montré Tiede (1999), une grammaire de Lambek peut produire des langages d'arbres non réguliers alors que les langages d'arbres produits par les grammaires hors contextes sont des langages d'arbres réguliers – produits par des automates d'arbres assez simples. Nous verrons ci-après (dans les sections 4, 5 et 6) des extensions du calcul de Lambek qui décrivent des langages légèrement contextuels tout en gardant la structure déductive du calcul de Lambek et la correspondance calculable avec la sémantique

– *Dérivations normales, propriété de la sous formule, décidabilité.* Toute dérivation du calcul de Lambek peut être normalisée, en une dérivation sans détour, un détour étant une introduction d'un connecteur immédiatement suivie d'une élimination du même connecteur. Pour chercher une dérivation, il suffit donc d'en chercher une normale. Or toute dérivation normale a une propriété très particulière : les formules (catégories) qui apparaissent dans une dérivation normale sont toutes présentes dans le séquent à dériver. Pour dériver un séquent – par exemple pour analyser une phrase – il suffit donc d'essayer les quatre règles possibles en prenant garde qu'elles ne fassent apparaître que des sous-catégories des catégories présentes dans le séquent, et il n'y en a qu'un nombre fini à explorer ; l'exploration de chacun de ces cas consiste à essayer de dériver un séquent contenant moins de symboles. Il s'ensuit que la dérivabilité d'un séquent, et donc l'analyse syntaxique, sont des questions décidables.

– *Complexité de l'analyse syntaxique.* La prouvabilité d'un séquent dans le calcul de Lambek en fonction du nombre de symboles du séquent à démontrer est un problème NP complet (Pentus, 2006) ; c'est un problème qui est résoluble en temps polynomial non déterministe et donc *a priori* exponentiel sur une machine standard. Si le degré d'imbrication des symboles « / » et « \ » est inférieur à k, l'analyse d'une phrase est polynomiale en fonction du nombre de mots, et le degré du polynôme dépend de l'entier k ; le lexique étant fini, le degré d'imbrication maximale est borné et l'analyse syntaxique elle-même est donc polynomiale en fonction du nombre de mots (Pentus, 2010). L'analyse peut également être rendue cubique en traduisant la grammaire de Lambek en une grammaire hors-contexte équivalente, mais la taille de la grammaire hors-contexte, qui est une constante multiplicative

du temps d'analyse, peut-être très grande. En pratique, la complexité de l'analyse dans les grammaires de Lambek ne pose pas de problème, car la complexité, dans le pire des cas, ne correspond pas à la réalité de l'analyse syntaxique de phrases réelles. Les problèmes d'ambiguïtés et du choix de la catégorie pour chaque mot sont des facteurs de complexité bien plus importants que de trouver une dérivation logique dans le calcul de Lambek. Le lecteur est renvoyé à l'article de Moot et Retoré (2016) pour une réflexion sur les mesures de complexité adaptées à l'analyse syntaxique et sémantique du langage naturel, en particulier dans le cadre des grammaires catégorielles.

– *Modèles.* Un modèle est la donnée des mots et des suites de mots des catégories atomiques 'S', 'sn', et 'n', qui sont notées [S], [sn], [n] et qui sont appelées l'interprétation, dans M, des catégories de base. Définissons l'interprétation des catégories complexes ainsi : (a\b) , respectivement (b/a), est l'ensemble des suites de mots, qui lorsqu'elles sont précédées (respectivement suivies) de n'importe quelle suite de mots de catégorie a, est toujours une suite de mots de catégorie b ; ceci n'est jamais que la traduction formelle du sens des catégories (a\b) et (b/a). Disons qu'un séquent $D_1,...,D_p \vdash X$ est vrai dans le modèle lorsque les suites de p suites de mots de $[D_1],...,[D_p]$ sont dans [X]. Le calcul de Lambek permet de dériver exactement les séquents vrais dans tout modèle. Le calcul de Lambek est la logique des suites de mots.

– *Le faux problème des ambiguïtés fallacieuses*. La littérature fait état d'un problème d'ambiguïtés fallacieuses dans les grammaires de Lambek, qui à notre avis n'a pas ou n'a plus de sens. Ce problème est la possibilité d'avoir des dérivations différentes dans le calcul de Lambek donnant la plupart des parenthésages syntaxiques possibles, y compris des parenthésages syntaxiquement erronés comme dans *'(Pierre (mange une)) pomme'*. Cette critique n'a pas lieu d'être ; il suffit, comme le font les analyseurs existants, de ne considérer que les dérivations normales (Tiede 1999). Reste alors un autre problème : si on souhaite identifier dérivation normale et structure syntaxique, certaines dérivations, certes similaires mais néanmoins différentes, conduisent à une même structure syntaxique, notamment à cause des permutations de règles. La bonne solution – celle utilisée par l'analyseur Grail – consiste à utiliser, pour représenter les dérivations, des graphes appelés réseaux de démonstrations, et cette fois, une analyse syntaxique correspond bien à une et unique structure syntaxique qui fait sens comme cela est expliqué dans Moot et Retoré (2012, chapitre 6).

3. CALCUL DE LAMBEK ET SÉMANTIQUE DE MONTAGUE

Jusqu'ici nous nous sommes contentés de l'aspect formel des grammaires catégorielles. Une question surgit : quel est leur apport pour la linguistique et le traitement automatique des langues ? Outre l'intérêt de la lexicalisation, il faut signaler leur interface aisée avec la sémantique de Montague. Ceci vient

de ce que les lambda-termes simplement typés sont des démonstrations en logique intuitionniste ; en effet l'isomorphisme de Curry-Howard (voir par exemple Girard 1988) montre que les lambda-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, celles-ci se plongent naturellement dans la logique intuitionniste. En effet, en lisant '(a / b)' et '(b / a)' comme 'a → b' (l'implication intuitionniste), chaque règle du calcul de Lambek est strictement analogue à une règle de la logique intuitionniste.

3.1. De l'analyse syntaxique à la représentation sémantique

La correspondance entre syntaxe et sémantique dans les grammaires de Lambek nécessite une brève présentation du lambda-calcul typé, ignoré des linguistes en dehors de la sémantique formelle. Nous nous limiterons au lambda-calcul typé avec deux types de base 'e' (individus) et 't' (propositions), celui qui permet de calculer de manière compositionnelle les formules logiques correspondant au sens des énoncés en sémantique formelle.

Les types sémantiques sont définis à partir de deux types de bases : 'e' qui englobe les entités ou individus et 't' qui regroupe les valeurs de vérités ou les propositions. À partir des types de base, on peut définir des types complexes, au moyen d'un unique connecteur → : 'A→B' est le type des fonctions de A dans B. Par exemple 'e→t' est le type d'un prédicat unaire (ou d'une partie des entités) : appliqué à une entité, on obtient une expression qui est vraie ou fausse. Par exemple « chaise » ou « bleu » ou « dort » sont de ce type, ce sont des prédicats à un argument. Le type 'e→(e→t)' est celui des prédicats à deux arguments, comme « regarde ». Les termes sont construits à partir de constantes, d'individus mais aussi de prédicats unaires, binaires, etc. Les constantes servent à construire les formules logiques (vus comme des lambda-termes) qui expriment le sens des énoncés : pour traduire logiquement *Tout barbier se rase lui-même*, il faut avoir un prédicat unaire 'barbier', un prédicat binaire 'rase', l'implication '⇒' et le quantificateur universel, d'où : ∀x (barbier(x) ⇒ rase(x,x)).

Pour écrire les formules logiques comme des lambda-termes typés, il faut des constantes logiques de deux sortes : certaines sont des connecteurs logiques qui servent à interpréter les opérateurs logiques de la langue, comme 'il existe' (∃), 'tout' (∀), 'implique' (⇒), 'et' (&) et d'autres sont les correspondants logiques référents des mots, comme 'rase (_,_)', et 'barbier (_)'.

Mot	Type
chat, souris	e→t
noir, gris	e→t
dort, court	e→t
regarde, mange, aime	e→(e→t)

Connecteur logique	Type
&	t→(t→t)
⇒	t→(t→t)
∀,∃	(e→t) →t

Les quantificateurs '∃' et '∀' méritent quelques commentaires : l'un comme l'autre, ils sont représentés par une constante logique de type (e→t)→t. Cela signifie qu'étant donnée une formule avec une variable libre de type 'e' (une propriété des entités), le quantificateur construit une proposition qui est vraie ou fausse : 'dort(x)' est une propriété avec une variable libre 'x' et '∀x (dort(x))' est une proposition, qui est vraie ou fausse. De même, '∃x (dort(x))' est une proposition qui est vraie ou fausse.

À partir d'un terme 'u' de type T et d'une variable 'x' (qui en général figure dans u) de type A, on peut définir le lambda-terme '$\lambda x^A.\ u^T$' de type A→T : c'est une fonction qui associe 'u' à 'x'. Le symbole 'λ' (lambda) sert à ne pas nommer la fonction : on écrit simplement '$\lambda x^A.\ u^T$' plutôt que :

f : A → T
x → u

À partir d'un terme 'f' de type A→T et d'un terme 'w' de type A, on peut définir un terme 'f(w)' de type T, qui est l'application de la fonction 'f' à 'w'. Par exemple :

- le lambda-terme '$(\lambda y^e\ .\ \lambda x^e\ .\ aime^{e\rightarrow e\rightarrow t}\ x^e\ y^e)$' est de type 'e→(e→t)' ;
- le lambda-terme '$(\lambda y^e\ .\ \lambda x^e\ .\ aime^{e\rightarrow e\rightarrow t}\ x^e\ y^e)\ Pierre^e$' est de type 'e→t' (c'est une propriété) ;
- le lambda-terme '$(\lambda y^e\ \lambda x^e\ aime^{e\rightarrow e\rightarrow t}\ x^e\ y^e)\ Pierre^e\ Marie^e$' est de type 't' (c'est donc une proposition).

Le quantificateur *tout* est analysé par '$\lambda P^{e\rightarrow t} \lambda Q^{e\rightarrow t} \forall\ \lambda x^e\ (\Rightarrow P(x)\ Q(x))$' : étant données deux propriétés P (par ex. *chat*) et Q (par ex. *dort*), ce lambda-terme construit la proposition « toute entité ayant la propriété P a la propriété Q » (ce qui correspond par ex. à *tout chat dort*). Le quantificateur *un* est analysé par '$\lambda P^{e\rightarrow t} \lambda Q^{e\rightarrow t} \forall\ \lambda x\ (\&\ P(x)\ Q(x))$' : étant donné deux propriétés P (par ex. *chat*) et Q (par ex. *dort*), ce lambda-terme construit la proposition « un individu ayant la propriété P a aussi la propriété Q » (ce qui correspond par exemple à *« un chat dort »*).

Des lambda-termes sont aisément obtenus à partir de dérivations similaires aux dérivations du calcul de Lambek ; ce sont des dérivations, appelées dérivations linéaires, dans lesquelles les symboles '/' et '\' deviennent '→'. Chaque formule X du contexte est étiquetée par une variable du lambda-calcul de ce même type X, et un lambda-terme ayant pour type la conclusion du séquent est construit au fur et à mesure de la dérivation :

$x : A \vdash x : A$

Lambda abstraction (à rapprocher de /i et \i du calcul de Lambek) :

$$\frac{x_1 : D_1,\ldots, x_p : D_p , x : A \vdash u : B}{x_1 : D_1,\ldots, x_p : D_p \vdash \lambda x.\, u : A\rightarrow B} \rightarrow /i$$

$$\frac{y_1 : G_1,\ldots, y_n : G_n \vdash w : A \qquad x_1 : D_1,\ldots, x_p : D_p \vdash f : A\rightarrow B}{y_1 : G_1,\ldots, y_n : Gn, x_1 : D_1,\ldots, x_p : D_p \vdash f(w) : B} \rightarrow e$$

Pour être tout à fait précis, il faut aussi une règle qui permet de permuter les hypothèses.

$$\frac{x_1 : D_1,\ldots, x_p : D_p \vdash u : X}{x_{\sigma(1)} : D_{\sigma(1)},\ldots, x_{\sigma(p)} : D_{\sigma(p)} \vdash u : X} \text{ échange}$$

Afin de clore cette représentation, mentionnons aussi la béta-réduction. Un lambda-terme (ou une partie d'un lambda-terme) de la forme '$(\lambda x^A.\ u^T)\ w^A$' se réduit à l'expression '$u^T[x^A := w^A]$', c'est-à-dire au terme u dans lequel toutes les occurrences de la variable 'x' ont été remplacées par 'w' ; on notera que 'x' et 'w' sont tous les deux de types A, sinon la substitution conduirait à des termes incorrectement typés. Cette réduction revient à dire que l'application d'une fonction '$(\lambda x^A.\ u^T)$' à un argument 'w^A' s'évalue en remplaçant les occurrences de la variable 'x^A' dans l'expression de la fonction, par l'argument 'w^A'. Hormis l'absence de nom des fonctions, c'est en tout point comparable aux mathématiques apprises au lycée : la fonction f : x /→ x^2-2x+1 qui est appliquée à l'argument (2a+b), c'est-à-dire f(2a+b), a pour valeur $(2a+b)^2$–2(2a+b)+1. Dans l'exemple ci-dessous, le lambda-terme :

'$(\lambda y^e\ \lambda x^e\ aime^{e\rightarrow e\rightarrow t}\ x^e\ y^e)\ Jacques^e\ Marie^e$'

se réduit à :

'$(\lambda x^e\ aime^{e\rightarrow e\rightarrow t}\ x^e\ Jacques^e)\ Marie^e$'

qui lui-même se réduit à :

'$aime^{e\rightarrow e\rightarrow t}\ Marie^e\ Jacques^e$'

Dans cette réduction, '$Jacques^e$' a pris la place de y^e, puis '$Marie^e$' celle de x^e. La béta-réduction se termine toujours avec un lambda-terme réduit avec les mêmes constantes que celles d'avant réduction. Les lambda-termes qui ne se réduisent plus sont dits normaux et les lambda-termes normaux de types 't' qui utilisent les constantes, comme celles que nous avons données précédemment, correspondent toujours à des formules logiques (écrites de façon non conventionnelle). On trouvera les explications complètes dans le chapitre 3 de Moot et Retoré (2012).

Commençons par définir une traduction des catégories syntaxiques vers les types sémantiques : ceux-ci sont les formules de la logique minimale (formulée avec le seul connecteur →, l'implication intuitionniste) construite sur les deux types 'e' (entités) et 't' (valeurs de vérité, propositions).

(catégorie syntaxique)*	=	**type sémantique**
S*	=	t — Une phrase est une proposition (qui peut être vraie ou fausse)
sn*	=	e — Un syntagme nominal est un individu
n*	=	e→t — Un nom commun est un prédicat
(b/a)* = (a\b)*	=	$a^* \rightarrow b^*$ — Étend la traduction aux catégories complexes.

À tout mot *m* de catégorie syntaxique 'c', le lexique va également associer un lambda-terme 'v' de sorte que le type de 'v' soit précisément '$(c)^*$', c'est-à-dire le « pendant sémantique » de la catégorie syntaxique 'c' de 'v' ; si un mot a plusieurs catégories syntaxiques, alors il aura aussi plusieurs lambda-termes sémantiques[4].

La sémantique de l'énoncé s'obtient par le simple algorithme suivant (qui sera illustré par le déroulement d'un exemple complet dans la section suivante car il serait inutile et inintelligible de définir précisément, ici, toutes les notions en toute généralité). Cet algorithme calcule les formules logiques représentant le sens de l'énoncé analysé à partir de :

– *la sémantique de chacun des mots* $m_1,\ldots, m_n$ c'est-à-dire des lambda-termes 'v_i' de type respectifs $(C_i)^*$;
– *une analyse syntaxique* d *de* $m_1,\ldots,m_n$ dans le calcul de Lambek, c'est-à-dire une dérivation d de $C_1,\ldots,C_n \vdash S$ où C_i est la catégorie syntaxique du mot m_i.

Les étapes de l'algorithme sont les suivantes :

1. Remplacer dans la dérivation syntaxique 'd' toute catégorie syntaxique 'c' par le type sémantique c^* ; on obtient alors la dérivation 'd^*', en logique linéaire intuitionniste, $(C_1)^*,\ldots,(C_n)^* \vdash t$. Les règles données

4 Ces constantes peuvent contenir des opérateurs d'intensionnalité, « ∧ » et « ∨ » mais pour simplifier nous les omettons ; en effet, ces opérateurs ne sont introduits que dans le lexique, ils ne sont pas perturbés par les règles, et ils n'influent pas sur l'algorithme d'interprétation des énoncés, tandis qu'une présentation de la logique intensionnelle dépasserait le cadre de cet article.

ci-dessus associent à ‘d^*’ un lambda-terme de type ‘t’, qui contient une variable x_i libre de type $(C_i)^*$ pour chaque mot m_i.
2. Remplacer dans ‘d^*’ chaque variable x_i par le lambda-terme v_i qui est également de type $(C_i)^*$.
3. Réduire le lambda-terme obtenu par béta réduction, ce qui fournit la représentation sémantique de l’énoncé analysé (une autre analyse du même énoncé peut fournir une représentation sémantique distincte). Ce lambda-terme est une écriture particulière d’une formule logique.

Il est possible de prouver que cet algorithme est correct et, en particulier, que le résultat est toujours une formule logique, comme le montre le chapitre 3 de Moot et Retoré (2012). Les représentations sémantiques ainsi obtenues sont des formules logiques qui précisent ce qui est affirmé, réfuté, supposé, etc.

Mais ce n’est qu’une partie du sens. Les connotations, les champs sémantiques impliqués ne sont pas pris en compte. Ainsi, *la chaise a aboyé* doit être rejetée car, au sens propre, seuls les chiens, ou assimilés, aboient. Pour ce faire, le lexique sémantique doit être enrichi par de nombreux types de base. Cependant, la phrase *« le sergent a aboyé après la nouvelle recrue »* doit pouvoir être interprétée. Il faut alors recourir à des transformations pour modéliser ces usages métaphoriques. Finalement, la phrase « *Blanche a fini mon livre* » s’interprétera comme « Blanche a fini une action relative à mon livre », sans pouvoir décider que cette action soit « lire » ou « écrire » ou encore « relier » le livre en question, voire le « manger » si « Blanche » est une chèvre. Ces raffinements lexicaux (Retoré 2014, Lafourcade *et al*. 2018) réussissent à calculer la sémantique de ce genre d’exemples même si l’acquisition des données nécessaire reste un défi.

3.2. Un exemple

Considérons le lexique syntaxique et sémantique que voici, dans lequel les prédicats du langage logique sont notés par les mots anglais correspondants, afin de ne pas confondre, par exemple « parler de » et le prédicat binaire correspondant, noté « talk about ».

mot	***catégorie syntaxique*** u ***type sémantique*** u^* ***semantics :*** λ***-terme de type*** u^* x^v *la variable ou constante* x *est de type* v
certains	$(S/(sn\backslash S))/n = C$ $(e \to t) \to ((e \to t) \to t) = C^*$ $\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\&^{t\to(t\to t)}(P\ x)(Q\ x))))$
énoncés	$n = E$ $e \to t = E^*$ $\lambda x^e(\texttt{statement}^{e\to t}\ x)$
parlent_de	$(sn\backslash S)/sn = P$ $e \to (e \to t) = P^*$ $\lambda y^e\ \lambda x^e\ ((\texttt{speak\_about}^{e\to(e\to t)}\ x)y)$
eux-mêmes	$((sn\backslash S)/sn)\backslash(sn/S) = X$ $(e \to (e \to t)) \to (e \to t) = X^*$ $\lambda P^{e\to(e\to t)}\ \lambda x^e\ ((P\ x)x)$

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 la conclusion :

((S/(sn\S))/n), n, ((sn\S)/sn), ((sn\S)/sn)\(sn\S)) ⊢ S

c'est-à-dire, avec les abréviations des unités lexicales : C, E, P, X ⊢ S :

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

En utilisant la transformation des types syntaxiques en type sémantiques, on obtient la dérivation linéaire suivante, où, comme indiqué dans le lexique supra C*, E*, P*, X* sont des abréviations des types sémantiques associés à C, E, P, X :

$$\dfrac{\dfrac{\mathsf{c}{:}C^* \vdash \mathsf{c}{:}(e\to t)\to(e\to t)\to t \quad \mathsf{e}{:}E \vdash \mathsf{e}{:}(e\to t)}{\mathsf{c}{:}C, \mathsf{e}{:}E \vdash (\mathsf{c}\ \mathsf{e}){:}(e\to t)\to t}\,/_e \quad \dfrac{\mathsf{p}{:}P \vdash \mathsf{p}{:}(e\to(e\to t)) \quad \mathsf{x}{:}X \vdash \mathsf{x}{:}(e\to(e\to t))}{\mathsf{p}{:}P, \mathsf{x}{:}X \vdash (\mathsf{x}\ \mathsf{p}){:}(e\to(e\to t))\to(e\to t)}\,\backslash_e}{\mathsf{c}{:}C, \mathsf{e}{:}E, \mathsf{p}{:}P, \mathsf{x}{:}X \vdash (\mathsf{c}\ \mathsf{e})(\mathsf{x}\ \mathsf{p}){:}t}\,/_e$$

C'est une dérivation linéaire telle qu'elle a été définie au paragraphe 3.2. Le lambda-terme qui code cette dérivation est, comme indiqué à la fin de la dérivation, '((c e) (x p))' de type 't' où c, e, x, p sont des variables de types respectifs C*, E*, P*, X*. Remplaçons ces variables par les lambda- termes ayant les mêmes types qui sont associés aux mots par le lexique, en omettant

toutefois les types à l'intérieur des C^*, E^*, P^*, X^* des lambda-termes, qui figurent dans le lexique ci-dessus. On obtient le lambda-terme suivant (écrit sur les deux premières lignes) de type 't', que l'on réduit :

$$\Big(\big(\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\&(P\ x)(Q\ x)))))\ \big(\lambda x^{e}(\texttt{statement}^{e\to t}\ x)\big)\Big)\ \Big(\big(\lambda P^{e\to(e\to t)}\ \lambda x^{e}\ ((P\ x)x)\big)\ \big(\lambda y^{e}\ \lambda x^{e}\ ((\texttt{speak\_about}^{e\to(e\to t)}\ x)y)\big)\Big)$$

$$\downarrow \beta$$

$$\big(\lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\&^{t\to(t\to t)}(\texttt{statement}^{e\to t}\ x)(Q\ x)))))\ \big(\lambda x^{e}\ ((\texttt{speak\_about}^{e\to(e\to t)}\ x)x)\big)$$

$$\downarrow \beta$$

$$\big(\exists^{(e\to t)\to t}\ (\lambda x^{e}(\&(\texttt{statement}^{e\to t}\ x)((\texttt{speak\_about}^{e\to(e\to t)}\ x)x)))\big)$$

Ce lambda-terme réduit, en se souvenant que *x* dans ce lambda-terme est de type e, représente la formule du calcul des prédicats :

(∃ x) (statement(x) & speak_about (x,x))

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

4. EXTENSIONS DES GRAMMAIRES DE LAMBEK

Pour résumer, les grammaires de Lambek ont pour avantages : leurs propriétés formelles, d'être lexicalisées, d'avoir, du moins en pratique, une complexité d'analyse raisonnable. Ces grammaires permettent surtout d'automatiser très simplement le calcul des représentations sémantiques ; signalons aussi qu'elles peuvent être apprises automatiquement à partir de corpus, mais nous n'en parlerons pas dans cet article ; le lecteur pourra, sur ce sujet, consulter Bonato et Retoré (2014), Bechet *et al.* (2007), ou Tellier (2006) qui utilise la correspondance avec la sémantique. Le principal défaut des grammaires de Lambek est leur incapacité à décrire certaines constructions syntaxiques complexes, et parfois de donner aux phrases une structure syntaxique non naturelle, plus proche de la structure sémantique, notamment pour les quantificateurs généralisés – par exemple l'analyse de « *Paul mange une pizza* » fait apparaître le prédicat « *Paul mange x* » qui n'est pas un constituant de la phrase. Nous allons donc présenter deux développements récents des grammaires catégorielles qui, l'un comme l'autre, servent à étendre leur couverture syntaxique, tout en préservant la simplicité de la correspondance avec l'interprétation sémantique. Schématiquement, il y a deux manières de procéder pour étendre les grammaires catégorielles, que nous allons présenter ci-après :

– étendre la logique par d'autres connecteurs, avec des règles et axiomes gérant ces nouveaux connecteurs ; c'est l'approche suivie par les grammaires multimodales de Moortgat et d'autres auteurs (voir par exemple (Moortgat 2010 ; Morril 2010) ;

– garder la logique simple du calcul de Lambek, avec les analyses syntaxiques « à la Lambek », et dériver à partir de cette structure syntaxique profonde *via* un autre processus, non logique, l'ordre des mots et la structure sémantique ; c'est ce que nous avons proposé dans (Amblard *et al* 2010) en lien avec la syntaxe minimaliste de Chomsky (1993) et sa formalisation par Stabler (1997).

Ces extensions sont loin d'être les seules. Des chercheurs ont souhaité étendre les grammaires de Lambek en se rapprochant d'autres formalismes, par exemple en se rapprochant des grammaires de dépendance. Ainsi, Alexandre Dikovsky a proposé les grammaires catégorielles de dépendance, enrichies par les connecteurs /* et *\ qui trouvent leurs arguments à une certaine distance inconnue à droite ou à gauche, ce qui est complexe lorsque les dépendances ne sont pas projectives. Cette extension est en fait plutôt une extension des grammaires AB et des grammaires de réécriture. On en trouvera une présentation complète dans l'article posthume de Dikovsky (Dekhtiar *et al*. 2015). Les grammaires d'interaction de Perrier (1999) sont également une extension des grammaires catégorielles « à la Lambek », inspirée par les grammaires de dépendance, dont est issue une grammaire à large échelle du français, FRIGRAM (Perrier et Guillaume, 2015). La logique est davantage présente dans ce style de grammaire que dans celles de Dikovsky, puisque la logique linéaire y est utilisée pour exprimer les valences syntaxiques de certains mots auxquels d'autres répondent ; c'est une forme de consommation de « ressources » que la logique linéaire décrit aisément. Les grammaires d'interaction contiennent néanmoins des contraintes sur l'ordre des mots qui échappent au calcul de Lambek et à la logique linéaire : ici, comme chez Dikovsky, la grammaire utilise la notion de « à une distance inconnue à droite » ou « à une distance inconnue à gauche ».

D'autres chercheurs ont cherché à se rapprocher des grammaires d'arbres adjoints. Pour ce faire, il est plus aisé d'associer à chaque mot des fragments de dérivations plutôt que des simples catégories, et ces dérivations sont alors assemblées pour obtenir une dérivation correcte et complète, dont sont ensuite dérivés la structure syntaxique et l'ordre des mots. Plusieurs travaux distincts ont réussi à représenter fidèlement les grammaires d'arbres adjoints avec des fragments de dérivation du calcul de Lambek (ou de *pomset logic*, une autre forme de logique linéaire non commutative) comme l'ont montré Abrusci *et al*. (1997), Joshi et Kulick (1997), Lecomte et Retoré (1998), Pogodalla (1999).

L'approche des grammaires catégorielles abstraites de De Groote est différente. Elle ressemble davantage à la sémantique de Montague, que nous avons présentée en section 3. Un premier lexique associant à chaque mot un

lambda-terme linéaire donne une première dérivation, qui est une analyse de la structure profonde. Dans ce lambda-terme sont insérés les lambda-termes d'un second lexique afin d'obtenir le codage dans le lambda-calcul de l'arbre syntaxique (la structure de surface). À la place du second lexique, on peut utiliser un autre lexique qui lui donnera le codage en lambda-calcul de la représentation sémantique (De Groote 2001). L'organisation grammaticale fonctionne avec deux niveaux : on dérive une structure profonde dont on dérive ensuite l'arbre syntaxique vu comme un lambda-terme, en insérant des lambda-termes issus du lexique dans le premier lambda-terme. C'est très comparable à ce que nous avons fait en section 3 : à partir de la structure syntaxique (un lambda-terme linéaire) nous avons calculé une formule logique (un autre lambda-terme) en insérant des lambda-termes issus du lexique. La dérivation sémantique procède de la même manière : à partir de la structure profonde on dérive une formule logique. Cette structure profonde n'est pas sans rappeler la structure dérivationnelle du minimalisme avec la structure intermédiaire *spell out* dont se dérivent d'une part, la forme phonétique avec les *overt move* et d'autre part, la forme logique avec les *covert move*.

5. GRAMMAIRES CATÉGORIELLES MULTIMODALES – LA PLATEFORME GRAIL

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 (1988, 1997), d'un côté et par Morrill (1994) et Hepple (1990), de l'autre.

5.1. Calcul non associatif et grammaires multimodales

Le système minimal considéré par Moortgat (1988, 2010) est le calcul non associatif de Lambek (1961), dans lequel la prouvabilité se décide en un temps polynomial en fonction de la taille du séquent ; l'analyse syntaxique est donc polynomiale en fonction du nombre de mots. 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 : on notera T[A,B,C] une structure de parenthèses sur les catégorie A, B, et C comme par exemple <<A,B>,C>. Les règles binaires précisent bien sûr comment les deux arbres binaires d'hypothèses se combinent en un seul arbre binaire. Donnons les règles du calcul de Lambek non associatif :

Règles d'élimination de / et \ avec des contextes arborescents :

$$\frac{T[G_1,\ldots,G_n] \vdash A \qquad T'[D_1,\ldots,D_p] \vdash A\backslash B}{\langle T[G_1,\ldots,G_n], T[D_1,\ldots,D_p]\rangle \vdash B}\ \backslash_e$$

$$\frac{T'[D_1,\ldots,D_p] \vdash B/A \qquad T[G_1,\ldots,G_n] \vdash A}{\langle T'[D_1,\ldots,D_p], T[G_1,\ldots,G_n]\rangle \vdash B} /_e$$

Règles d'introduction de / et \ avec des contextes arborescents :

$$\frac{\langle A, T[D_1,\ldots,D_p]\rangle \vdash B}{T[D_1,\ldots,D_p] \vdash A\backslash B} \backslash_i \qquad \frac{\langle T[D_1,\ldots,D_p], A\rangle \vdash B}{T[D_1,\ldots,D_p] \vdash B/A} /_i$$

Il est alors impossible, dans un tel calcul, de démontrer la règle de composition (A/B, B/C) ⊢ A/C, ainsi que l'associativité que l'on peut voir comme ((X, ((X\U)/Z)), Z) ⊢ U, d'où le nom calcul non associatif. On peut aussi rendre compte des îlots, dont on ne peut extraire un constituant ; la phrase *La mouette que Gaston aime et Lebrac déteste* t_i ... est reconnue tandis que la phrase suivante ***La mouette que Gaston aime Julie et Lebrac déteste*** t_i ... ne l'est pas ; dans le calcul de Lambek associatif que nous avons présenté, si on procède avec les catégories usuelles, cette phrase est reconnue, mais à tort.

L'idée de base des grammaires multimodales est d'étendre ce calcul minimal dans deux directions :

– en traitant, dans un même calcul logique, de plusieurs familles de connecteurs $/_i$ et $\backslash_i$ 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 ;

– en incluant également des modalités, $\Box_i$ et $\Diamond_i$, 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 composition 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. Certaines règles lient modalités et connecteurs binaires : ces règles sont exprimées soit par des restrictions sur la forme des séquents prémisses du calcul des séquents, 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 qui permettent de contrôler la duplication des hypothèses qui est prohibée dans le système de base. Moortgat et d'autres auteurs ont alors formulé des théorèmes de plongements fidèles des systèmes contraints dans les systèmes

moins contraints à l'aide de ces modalités structurelles. (Kurtonina et Moortgat 1997)

En utilisant ces connecteurs multiples des grammaires multimodales, Moot et Retoré (2006) ont pu rendre compte des clitiques du français et de leur montée ou non en présence d'auxiliaires : *Je la lui fais réparer / Je te laisse la réparer.*

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 chomskyennes (Pollock 1997). On arrive ainsi à dépasser les limites décrites à la fin de la section 2.1. 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, que [Spirou a vu* t_i *hier]…*) ;

– les dépendances croisées des complétives en néerlandais.

L'analyse syntaxique dans ce genre de systèmes est décidable, et les langages reconnus sont les langages contextuels. L'analyse syntaxique est polynomiale pour des systèmes restreints qui suffisent à engendrer des fragments raisonnables du langage naturel. Par exemple, un système restreint de grammaires catégorielles multimodales correspond aux grammaires d'arbres adjoints comme cela a été établi par Moot (2002).

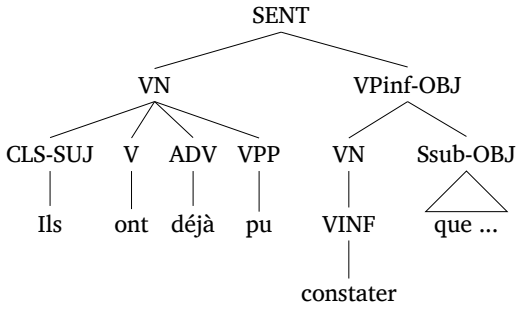
La construction de la sémantique de Montague présentée dans le calcul de Lambek s'étend parfaitement à ces systèmes enrichis, sans étendre le lambda-calcul (sans quoi celui-ci ne correspondrait plus au calcul des prédicats). Les modalités sont sans effet sur la sémantique, et les différentes variantes des connecteurs de Lambek sont interprétées comme le sont / et \. C'est heureux car la correspondance avec la sémantique est le principal attrait des grammaires de Lambek.

Morrill (1994) 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. Par la suite, Morrill (2010) a considérablement enrichi ce genre de calcul afin de rendre compte des constituants discontinus, au prix d'un système logique dont certaines règles sont très complexes.

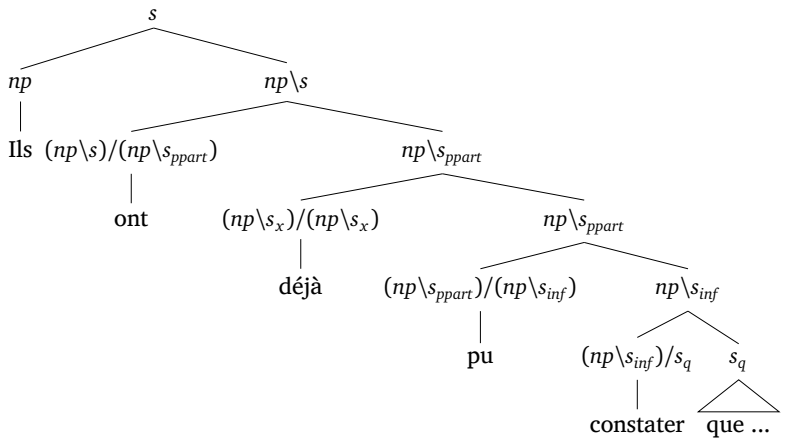
5.2. La plateforme GRAIL d'analyse syntaxique et sémantique du français à large échelle

Richard Moot (2010, 2015) a développé depuis 15 ans un analyseur du français à large échelle, après avoir construit grammaire pour le néerlandais parlé (voir Moot 1998, 2002). La grammaire utilisée est une grammaire catégorielle multimodale acquise à partir de corpus. L'acquisition automatique est une tâche complexe, elle a été effectuée en convertissant les annotations du corpus *French Treebank* de 423 152 mots (corpus du *Monde* développé à Paris 7). Le lexique grammatical comporte 28 753 mots, avec jusqu'à 80 catégories par mot (*et*, *est*, ainsi que les prépositions reçoivent de nombreuses catégories).

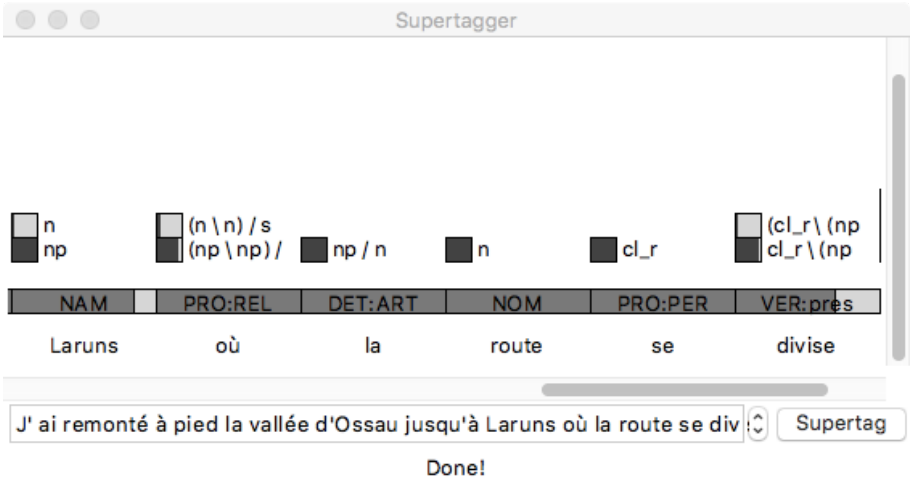
Arbre annoté de racine SENT=Sentence=Phrase
(tel qu'il apparait dans le French Tree Bank) :



Le même arbre est étiqueté par les catégories des grammaires multimodales :



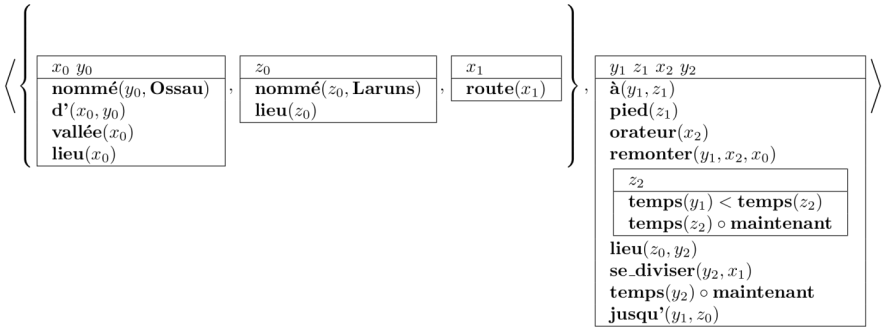
Une fois la grammaire acquise, l'analyse syntaxique commence par une étape préalable appelée *super tagging*. Il s'agit d'associer à chaque mot une catégorie « probable », de sorte que la suite de catégories ainsi obtenue pour l'ensemble de la phrase soit assez probable. Le *super tagger* utilisé est celui de Currant *et al*. (2006) qui ordonne, par probabilité décroissante, les catégories à utiliser pour chaque mot de la phrase. Dans la figure suivante qui présente le résultat du super-tagger pour la phrase analysée, la proportion de gris foncé, dans le carré associé à une catégorie, correspond à sa probabilité :



Puisqu'une analyse syntaxique, dans une grammaire catégorielle déductive, est une déduction logique, l'analyseur est un démonstrateur automatique pour la logique multimodale décrite en section 5.1. La phrase est analysée pour chacune des suites de catégories probables obtenues par le *super tagging* afin d'obtenir la bonne analyse sélectionnée parmi celles qui sont réalisées (en pratique les 7 meilleures sont analysées). La catégorie S_{main} désigne une proposition principale. En pratique, il y a beaucoup plus de catégories de base que les trois catégories *S, sn* et *n* utilisées dans la présentation des grammaires de Lambek. La déduction qui suit est produite par le système, y compris son écriture en LaTeX : c'est relativement illisible, mais cela montre à quoi ressemble une dérivation réelle, laquelle est faite pour être traitée par un système informatique et non pour être lue par un être humain.

$$\dfrac{\dfrac{\text{jusqu'}}{(s_{main}\backslash_1 s_{main})/pp_{\grave{a}}}\,[Lex] \quad \dfrac{\dfrac{\grave{a}}{pp_{\grave{a}}/n}\,[Lex] \quad \dfrac{\dfrac{\text{Laruns}}{n}\,[Lex] \quad \dfrac{\dfrac{\text{où}}{(n\backslash n)/s_{main}}\,[Lex] \quad \dfrac{\dfrac{\dfrac{\text{la}}{np/n}\,[Lex] \quad \dfrac{\text{route}}{n}\,[Lex]}{\text{la} \circ \text{route} \vdash np}\,[/E] \quad \dfrac{\dfrac{\text{se}}{cl_r}\,[Lex] \quad \dfrac{\text{divise}}{cl_r\backslash(np\backslash s_{main})}\,[Lex]}{\text{se} \circ \text{divise} \vdash np\backslash s_{main}}\,[\backslash E]}{(\text{la} \circ \text{route}) \circ (\text{se} \circ \text{divise}) \vdash s_{main}}\,[/E]}{\text{où} \circ ((\text{la} \circ \text{route}) \circ (\text{se} \circ \text{divise})) \vdash n\backslash n}\,[\backslash E]}{\text{Laruns} \circ (\text{où} \circ ((\text{la} \circ \text{route}) \circ (\text{se} \circ \text{divise}))) \vdash n}\,[/E]}{\text{à} \circ (\text{Laruns} \circ (\text{où} \circ ((\text{la} \circ \text{route}) \circ (\text{se} \circ \text{divise})))) \vdash pp_{\grave{a}}}\,[/E]}{\text{jusqu'} \circ (\text{à} \circ (\text{Laruns} \circ (\text{où} \circ ((\text{la} \circ \text{route}) \circ (\text{se} \circ \text{divise}))))) \vdash s_{main}\backslash_1 s_{main}}$$

La sémantique de Montague permet ensuite d'obtenir des représentations sémantiques sous la forme de structures discursives, ce qui permet notamment de résoudre les anaphores – à condition de disposer d'un lexique sémantique. Celui-ci a été obtenue au moyen de 350 règles par défaut, complétées par 500 entrées spécifiques, pour les mots ayant un fort impact sur la structure logique de la phrase, notamment les coordinations et les quantificateurs. Le système produit directement la représentation sémantique d'une phrase vue comme une structure discursive (DRS) – y compris sa mise en page de la DRS avec les boîtes et les accolades. Là encore, il s'agit d'une représentation en machine qui peut être interrogée et dont on peut déduire des conséquences, mais qui n'est pas vraiment faite pour être lue par un utilisateur humain.



Une difficulté apparaît puisqu'il s'agit d'acquérir les subtilités de la sémantique lexicale, dont nous avons parlé à la fin de 3.2., afin de choisir le sens adéquat en tenant compte du contexte. L'approche choisie est celle de MGL présentée par Retoré (2014) : plutôt que d'acquérir les sens métaphoriques et les contextes qui autorisent ces glissements de sens, nous envisageons d'extraire ces données de réseaux lexicaux existants, comme cela est expliqué par dans Lafourcade *et al*. (2018).

6. GRAMMAIRES MINIMALISTES CATÉGORIELLES – CONVERGENCES THÉORIQUES

La convergence entre grammaires catégorielles et programme minimaliste de la grammaire générative a été notée dès 1995 (Epstein et Berwick 1995), ce qui nous a conduits à proposer une synthèse issue de cette convergence (Retoré et Stabler 2004). D'une part, ces deux visions de la syntaxe sont lexicalisées (« *language variation is purely lexical* ») et d'autre part, l'opération de composition simple MERGE rappelle les règles des grammaires AB. Le calcul de la forme logique dans les grammaires minimalistes

est bien plus obscur que dans les grammaires catégorielles, mais les grammaires minimalistes offrent une richesse syntaxique au-delà de celle des grammaires catégorielles. C'est pour cela que nous avons essayé de rendre compte des structures syntaxiques minimalistes dans un cadre catégoriel, tout en préservant la facilité du calcul de l'interprétation sémantique. Afin de donner une formalisation logique du programme minimaliste de la grammaire générative, nous sommes partis de la formalisation, due à Stabler (1997), des grammaires minimalistes, celle-ci propose une grammaire lexicalisée où les mots sont associés à des suites de traits qui déclenchent et régissent les opérations MERGE (binaire) et MOVE (unaire) qui s'appliquent à des mots et à des arbres syntaxiques déjà construits. Les suites de traits sont traduites par des catégories « à la Lambek » (avec le produit commutatif en plus des deux connecteurs / et \) afin d'obtenir un lexique catégoriel, qui, réciproquement, peut être traduit en un lexique minimaliste. La difficulté est bien évidemment de rendre compte, dans un système de dérivations logiques, des déplacements (MOVE). Pour ce faire, nous avons proposé un système qui, à partir d'un lexique, procède en deux étapes :

1°) Une dérivation de conclusion S, dans une extension du calcul de Lambek ; en plus de / et \ (les deux implications), il y a un produit • (une conjonction) satisfaisant A•B = B•A.

2°) Un étiquetage de cette dérivation donne l'ordre des mots, le produit commutatif induisant un déplacement de matériel phonétique.

Un autre étiquetage permet de calculer la sémantique de Montague, ou des variantes en DRT (théorie des représentations discursives), voire, comme dans Amblard *et al*. (2010), qui reprend nos travaux sur ce sujet, d'avoir des représentations sémantiques sous la forme de lambda-mu DRS qui se réduisent en l'une des lectures possibles, en cas d'ambiguïtés de portée des quantificateurs.

7. CONCLUSION

Après une présentation des grammaires catégorielles comme des systèmes déductifs, et plus particulièrement du calcul de Lambek, nous avons succinctement présenté les extensions de ces formalismes déductifs qui permettent d'étendre la couverture syntaxique des grammaires de Lambek, tout en préservant la correspondance entre la structure syntaxique, obtenue par une grammaire catégorielle, avec l'interprétation sémantique de la phrase analysée. Les perspectives ouvertes par ce genre de formalisme nous semblent être à la fois théoriques et pratiques.

En pratique, il nous semble important de développer des analyseurs syntaxiques et sémantiques profonds, dans la lignée de l'analyseur *Grail* du français à large échelle. Le lexique syntaxique peut être automatiquement acquis sur des corpus annotés. Les lambda-termes sémantiques peuvent aussi l'être, mais, lorsqu'une analyse sémantique subtile est requise, il y a une

réelle difficulté d'acquérir le sens lexical, le lambda-terme associé au mot, ou plus difficile encore, les coercitions qui choisissent le sens en fonction du contexte. Même avec les progrès de l'Intelligence Artificielle par *machine learning*, il semble difficile d'acquérir, à partir de corpus, les données dont l'analyseur syntaxique et sémantique a besoin. Une piste peut être suivie : extraire ces informations à partir des réseaux lexicaux existants comme *Rezo*, le réseau lexical du français construit par les internautes au moyen de jeux sérieux (Lafourcade *et al*. 2018).

En théorie, le point de vue logique des grammaires catégorielles questionne les liens qui unissent logique et langage. Initialement définies en opposition aux grammaires transformationnelles et à leurs éléments vides, les grammaires déductives se sont aujourd'hui fortement rapprochées du programme minimaliste de la grammaire générative. Les grammaires catégorielles permettent aujourd'hui de formaliser des notions comme la forme logique, l'apprentissage de la grammaire à partir d'exemples positifs ou de formuler l'universalité de certains principes grammaticaux.

La formalisation est également un moyen de confirmer ou d'infirmer les hypothèses proposées par les théories linguistiques notamment en termes de complexité (au sens informatique) et d'économie (au sens chomskyen). Par exemple, certaines structures syntaxiques sont supposément préférées à d'autres parce qu'elles sont plus économiques (par exemple *shortest move*) : leur formalisation catégorielle est-elle moins complexe (en nombre de règles, dans la taille des catégories impliquées, ...) ? Certaines lectures d'une même phrase sont supposées plus complexes (et donc moins probables) que d'autres lectures de la même phrase ; les grammaires catégorielles permettent de prédire certaines de ces préférences d'analyse. (Mirzapour *et al.* 2018).

RÉFÉRENCES

ABEILLÉ A. (1993). *Les nouvelles syntaxes*. Paris : Armand Colin.

ABRUSCI V.M., FOUQUERE C., VAUZEILLES J. (1997). Tree adjoining grammar and non-commutative linear logic. In : Retoré (ed.) (1997), 13-17.

AJDUKIEWICZ K. (1935). Die syntaktische konnexität. *Studia Philosophica* 1, 1-27.

AMBLARD M., DE GROOTE PH., POGODALLA S., RETORÉ CH. (eds) (2016). Logical Aspects of Computational Linguistics. *Celebrating 20 Years of LACL* (1996–2016), LNCS 10054 Springer.

AMBLARD M., LECOMTE A., RETORÉ CH. (2010). Categorial Minimalist Grammar : From Generative Syntax To Logical Form. *Linguistic Analysis* 36 (1-4), 273-308.

BAR-HILLEL Y. (1953). A quasi arithmetical notation for syntactic description. *Language* 29, 47-58.

BECHET D., BONATO R., DIKOVSKY A., FORET A., LE NIR Y., MOREAU E., RETORÉ CH., TELLIER I. (2007). Modèles algorithmiques de l'acquisition de la syntaxe : concepts et méthodes, résultats et problèmes. *Recherches Linguistiques de Vincennes* 36, 123-152.

BLACKBURN P., DYMETMAN M., LECOMTE A., RANTA A., RETORÉ C., VILLEMONTE DE LA CLERGERIE E. (1997). Logical aspects of computational linguistics : an introduction. In : Retoré (1997), 1-20.

BONATO R., RETORÉ CH. (2014). Learning Lambek Grammars from Proof Frames. In : C. Casadio, B. Coecke, M. Moortgat, P. Scott (eds), *Categories and types in logic, language and physics – Essays dedicated to Jim Lambek on the occasion of his 90th birthday*. Berlin : Springer, LNCS 8222.

CHOMSKY N. (1995). *The minimalist program*. Cambridge, MA : MIT Press.

CURRAN J., CLARK S., VADAS D. (2006). Multi-tagging for lexicalized-grammar parsing. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 697-704

DE GROOTE PH. (2001). Towards Abstract Categorial Grammars. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, 252-259.

DEKHTYAR, M., DIKOVSKY, A., KARLOV, B. (2015). Categorial dependency grammars. *Theoretical Computer Science*, 579, 33-63.

EPSTEIN S., BERWICK R. (1995). On the Convergence of 'Minimalist' Syntax and Categorial Grammar. In : A. Nijholt, G. Scollo and R. Steetkamp, (eds), *Algebraic Methods in Language Processing*.

GIRARD J.Y. (1987). Linear logic. *Theoretical Computer Science* 50(1), 1-102.

GIRARD J.Y., LAFONT Y., TAYLOR P. (1988). *Proofs and Types*. Cambridge University Press.

GOLD E.M. (1967). Language identification in the limit. *Information and control* 10, 447-474.

HEPPLE M.(1990). *The Grammar and Processing of Order and Dependency, a categorial approach*. PhD thesis, Centre of Cognitive Science, Edinburgh.

JOSHI A., KULICK S. (1997). Partial proof trees, resource sensitive logics and syntactic constraints. In : Retoré (1997), 21-42.

JOSHI A., SCHABES Y. (1997). Tree adjoining grammars. In : G. Rozenberg and A. Salomaa (eds), *Handbook of Formal Languages*, vol. 3, chapter 2. Springer.

JOSHI A., VIJAY-SHANKER K., WEIR D. (1991). The convergence of mildly context-sensitive grammar formalisms. In : P. Sells, S. Schieber and T. Wasow, (eds), *Fundational issues in natural language processing*. MIT Press.

KURTONINA N., MOORTGAT M. (1997). Structural control. In : P. Blackburn and M. de Rijke (eds), *Specifying Syntactic Structures*, 75-113, CSLI., Cambridge University Press.

LAFOURCADE M., MERY B., MIRZAPOUR M., MOOT R., RETORÉ CH, (2018). Collecting Crowd-Sourced Lexical Coercions for Compositional Semantic Analysis. In : S. Arai, K. Kojima, K. Mineshima, D. Bekki, K. Satoh, Y. Ohta (eds), *New Frontiers in Artificial Intelligence*, LNCS 10838, 214-230.

LAMBEK J. (1958). The mathematics of sentence structure. *American mathematical monthly* 65, 154-169.

LAMBEK J. (1961). On the calculus of syntactic types. In : R. Jakobson (ed.), *Structure of language and its mathematical aspects*. American Mathematical Society, 166-178.

LECOMTE A. (1996). Grammaire et théorie de la preuve : une introduction. *Traitement Automatique des Langues* 37(2), 1-38.

LECOMTE A., RETORÉ CH. (1998). Words as modules : a lexicalised grammar in the framework of linear logic proof nets. In : Carlos Martin-Vide (ed.), *Mathematical and Computational Analysis of Natural Language – selected papers from ICML`96, Studies in Functional and Structural Linguistics*, vol. 45, 129-144, John Benjamins.

MIRZAPOUR, M., PROST, J.-PH., RETORÉ, C. (2018). Categorial Proof Nets and Dependency Locality : A New Metric for Linguistic Complexity. In : K. Angelov, K. Liefke, R. Loukanova, M. Moortgat, S. Tojo (eds), *Proceedings of the Symposium on Logic and Algorithms in Computational Linguistics 2018*, Stockholm University, 73-86.

MONTAGUE R. (1974). *The collected papers of Richard Montague, edited and with an introduction by Richmond Thomason*. New Haven : Yale University Press.

MOORTGAT M. (1988). *Categorial Investigations*. Dordrecht : Foris.

MOORTGAT M. (2010). Categorial type logic. In : van Benthem and ter Meulen (eds), (2010), chapter 2, 95-180.

MOOT R (1998). Grail : An automated proof assistant for categorial grammar logics. In : R.C. Backhouse (ed.), *Proceedings of the 1998 User Interfaces for Theorem Provers Conference*, 120-129.

MOOT R. (2002). *Proof nets for linguistic analysis*. PhD Thesis. Universiteit Utrecht.

MOOT R. (2010). Wide-Coverage French Syntax and Semantics using Grail. *TALN 2010*.

MOOT R. (2015). A type-logical treebank for French. *Journal of Language Modelling* 3 (1), 229-264.

MOOT R., RETORÉ CH. (2006). Les indices pronominaux du français dans les grammaires catégorielles. *Linguisticæ Investigationes* 29, 137-146.

MOOT R., RETORÉ CH. (2012). *The logic of categorial grammars : a deductive account of natural language syntax and semantics*. LNCS 68 50, Springer.

MOOT R., RETORÉ CH. (2016). Natural Language Semantics and Computability. To appear in the *Journal of Logic, Language and Information*.

MORRILL G. (1994). *Type Logical Grammar*. Kluwer Academic Publishers.

MORRILL G. (2010). *Categorial Grammar : Logical Syntax, Semantics, and Processing*. Oxford University Press.

PARTEE B. (2010). Montague grammar. In : van Benthem and ter Meulen (eds) (2010), chapter 1, 5–92.

PENTUS M. (1993). Lambek grammars are context-free. *Logic in Computer Science*. IEEE Computer Society Press.

PENTUS, M. (2006). Lambek calculus is NP-complete. *Theoretical Computer Science* 357(1), 186-201

PENTUS, M. (2010). A polynomial-time algorithm for Lambek grammars of bounded order. *Linguistic Analysis* 36 (1-4), 441-471.

PERRIER G. (1999). Labelled proof nets for the syntax and semantics of natural languages. *Journal of the IGPL* 7(5), 629-654.

PERRIER G., GUILLAUME B. (2015). FRIGRAM : a French Interaction Grammar. *Journal of Language Modelling* 3(1), 265-316.

PINKER S. (1995). Language acquisition. In : L. Gleitman, M. Liberman and D.N. Osherson (eds), *An invitation to cognitive science*, chapter 6, 135-182.

POGODALLA S. (1999). LTAGs and pomset logic. In : M. Moortgat (ed), *Logical Aspects of Computational Linguistics*, LACL`98, selected papers, LNCS/LNAI. Springer-Verlag.

POLLOCK J.Y. (1997). *Langage et cognition : le programme minimaliste de la grammaire générative*. Paris : Presses Universitaires de France.

RETORÉ CH. (1996). Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues* 37(2), 39-70.

RETORÉ C. (ed) (1997). *Logical Aspects of Computational Linguistics*. LACL`96, vol. 1328, LNCS/LNAI, Springer.

RETORÉ CH. (2001). Systèmes déductifs et traitement des langues : un panorama des grammaires catégorielles. *Technique et Science Informatiques (TSI)* 3, 11-46.

RETORÉ CH. (2014). The Montagovian Generative Lexicon ΛTyn : a Type Theoretical Framework for Natural Language Semantics. In : R. Matthes and A. Schubert (eds), *19th International Conference on Types for Proofs and Programs* (TYPES 2013), LIPICS, vol. 26, 202-229.

RETORÉ CH., STABLER E. (2004). Generative grammars in resource logics. *Research on Language and Computation* 2(1), 3-25.

ROORDA D. (1991). *Resource logic : proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam.

ROZENBERG G., SALOMAA A., (eds) (1997). *Handbook of formal Language theory*. Springer.

STABLER E. (1997). Derivational minimalism. In : Retoré (1997), 68-95.

STEEDMAN M. (1988). Combinators and grammars. In : R. Oehrle, E. Bach, and D. Wheeler (eds), *Categorial Grammars and Natural Language Structures*. Dordrecht : Reidel.

TELLIER I. (2006). *Modéliser l'acquisition de la syntaxe du langage naturel via l'hypothèse de la primauté du sens*. Thèse d'Habilitation à diriger des recherches. Université de Lille 3.

TIEDE H.J. (1999). *Deductive Systems and Grammars : Proofs as Grammatical Structures.* PhD thesis, Indiana University.

VAN BENTHEM J. (1991). Language in Action : Categories, Lambdas and Dynamic Logic. *Studies in logic and the foundation of mathematics*, vol. 130, Amsterdam : North-Holland.

VAN BENTHEM J., TER MEULEN A. (eds) (2010). *Handbook of Logic and Language*, 2nd edition. Elsevier MIT Press.