MATHÉMATIQUES ET INFORMATIQUE

Les mathématiques de la linguistique computationnelle

Deuxième volet : Logique[1]

Christian Retoré[2]

À mon père et professeur de mathématiques de première, aujourd'hui artiste peintre, pour ses 70 ans

Cet article est le second volet d'une présentation de la linguistique computationnelle qui place l'accent sur les modèles mathématiques ou informatiques utilisés ou initiés par des considérations linguistiques. Nous avons organisé ces modèles en deux familles, non disjointes : la théorie des langages qu'a présentée le premier volet et la logique qui fait l'objet de ce second volet. Partant de remarques sur les origines communes de la logique et de la grammaire, on verra comment les grammaires catégorielles, en particulier depuis le calcul syntaxique de Joachim Lambek, font des dérivations syntaxiques des preuves dans une version non commutative de la logique linéaire de Jean-Yves Girard. La combinatoire particulière des grammaires catégorielles permet aussi de les acquérir automatiquement à partir d'exemples de phrases. Surtout, leur lien fort avec la théorie des types et le lambda calcul leur permet d'analyser la syntaxe des phrases (comme le fait la théorie des langages), mais aussi de calculer le sens qu'elles véhiculent. Finalement nous abordons les premiers résultats qui relient les approches décrites dans ces deux volets, la théorie des langages et la logique.

1. Logique et grammaire : une longue histoire

Tandis que le premier volet de notre présentation [60] s'est concentré sur des modèles calculatoires de la syntaxe de la phrase, celui-ci portera davantage vers le sens de la phrase en s'ancrant dans la tradition logique. En effet, depuis l'antiquité gréco-romaine (Aristote, Denis de Thrace) grammaire et logique ont un cheminement parallèle [35, 8], qui s'est poursuivi au Moyen-Âge (scholastique) [35, 14], au

[1] Je remercie de leurs relectures Christian Bassac (CLLE et INRIA Bordeaux Sud-Ouest, université de Bordeaux), Alain Lecomte (SFL, université Paris VIII), Jean-Jacques Loeb (LAREMA, université d'Angers), Frédéric Patras (CNRS, université de Nice), Sylvain Pogodalla (LORIA, INRIA Nancy Grand Est), Jean-Xavier Rampon (LINA, université de Nantes), Gilles Zémor (IMB, université de Bordeaux).

[2] LaBRI (CNRS et université de Bordeaux) & INRIA Bordeaux Sud-Ouest

XVIII^{e} (Port-Royal) [6, 5]... La raison en est simple : une phrase a une structure logique, la phrase simple est pour ainsi dire la version grand public de la proposition logique et la connexion entre logique et langage est toujours d'actualité [67, 26].

Un exemple que nous traiterons en entier plus loin est le suivant :

(1) Certains énoncés parlent d'eux-mêmes[3].

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

Ce parallèle entre phrases et propositions participe d'une plus vaste correspondance entre rôles logiques et catégories syntaxiques. Plutôt qu'une similitude établie *a posteriori*, c'est tout simplement que les notions, logiques et grammaticales, ont été développées conjointement.

Les groupes nominaux correspondent aux individus de la logique (individus ou variables d'individus souvent quantifiées), les verbes intransitifs et les groupes verbaux correspondent à des prédicats monadiques (unaires) tandis que les verbes transitifs correspondent aux prédicats binaires et les verbes ditransitifs (*quelqu'un donne quelque chose à quelqu'un*) à des prédicats ternaires. Les adjectifs partagent des caractères avec les noms et groupes nominaux (accord, déclinaisons) mais aussi avec les verbes puisqu'ils expriment un prédicat ; quant aux groupes prépositionnels, ce ne sont ni des prédicats, ni des individus.

Bien évidemment les débats de l'antiquité et de la scholastique étaient considérablement entravés par l'absence de formalisme logique et en particulier de variables, ainsi que par l'absence de logique d'ordre supérieur qui permette de quantifier sur les propriétés. Les adjectifs comme *bon* sont plutôt des prédicats de prédicats, tandis que les adjectifs comme *français* dits adjectifs intersectifs sont de simples prédicats. C'est ce qui explique que de (3) on puisse déduire (4) et pas (5).

(3) tous les médecins sont des conducteurs

(4) (donc) tous les médecins français sont des conducteurs français

(5) *(donc) tous les bons médecins sont des bons conducteurs

Cette origine commune étant posée, on remarquera que nous avons déjà croisé, dans le premier volet, certains domaines de la logique et en particulier de la logique mathématique. Plus précisément, il s'agissait de deux domaines fort distants dans l'espace de la logique mathématique.

D'une part, concernant la syntaxe du langage naturel et les arbres d'analyse nous avons évoqué la théorie des modèles qui permet de décrire la classe des modèles d'une théorie, notamment les structures d'arbres décrivant la structure syntaxique d'une phrase.

D'autre part nous avons évoqué, de manière moins formelle, dans les différents champs d'étude de la linguistique, la sémantique. Cette dernière étudie le « sens » véhiculé par la langue et notamment le sens d'une phrase ou d'un discours, et c'est de cette étude, initialement philosophique, qu'est issue la logique. La logique mathématique apparue au début du XX^{e} siècle notamment avec Frege est encore très proche des considérations linguistiques initiales. Il s'agit d'une part d'interpréter un énoncé dans un univers donné, mais aussi de construire à partir de ce qui est dit une représentation du sens, en général par une formule logique.

[3] Au sens de : « certains énoncés sont circulaires », comme *Cette phrase est vraie*, ou pire, *Je mens.*

A priori, ces deux aspects relèvent plutôt de la théorie des modèles : il s'agit d'interpréter une formule ou une théorie, de la relier aux structures dans lesquelles elle est vraie.

En fait, on peut unir ces deux aspects, forme et sens, par le biais d'un autre aspect de la logique, la théorie des types qui fait partie de la théorie de la démonstration. Ces théories étudient et formalisent le raisonnement logique, qu'il soit naturel ou mathématique, et les structures formelles utilisées par ce domaine de la logique, arbres et λ-termes, s'avèrent d'une pertinence qui excède largement la description du raisonnement : c'est aussi un modèle du calcul qui fonde aussi bien la programmation fonctionnelle que le calcul parallèle [28, 37, 43].

Ce rapprochement peut être *grosso modo* formulé ainsi : pourquoi ne pas identifier la structure d'une phrase avec la preuve formelle qu'elle est une phrase ? Il s'agira bien sûr d'une logique simple et élégante, bien adaptée à décrire une grammaire formelle, le calcul de Lambek [39]. Dans ce cas, on peut espérer traduire les formules particulières qui expriment la correction grammaticale de la phrase en formules de la logique ordinaire qui expriment le sens de la phrase, la preuve de bonne formation syntaxique se muant en une preuve de bonne formation sémantique : une phrase doit se traduire en une proposition.

On réalisera ainsi formellement un modèle en accord avec la compositionnalité chère à Gottlob Frege, même si cela est parfois discuté [35, 33] : le sens du tout est fonction du sens des parties. Richard Montague alla jusqu'à un parallèle règle à règle entre structure syntaxique et structure sémantique où la fonction qui compose les sens des parties est l'image de la composition syntaxique [65].

Nous allons consacrer l'essentiel de ce second volet aux grammaires catégorielles dites de Lambek, qui réalisent ce parallèle entre composition syntaxique et composition des sens. On verra qu'elles ramènent la bonne formation syntaxique d'une phrase à la prouvabilité d'une formule (ou au typage des constituants de la phrase) dans une logique. Tandis que la structure syntaxique est la preuve elle-même, un morphisme entre systèmes logiques transforme formules et preuves en représentations sémantiques. Cette organisation de la grammaire permet aussi de l'acquérir à partir d'exemples de phrases en un nombre fini d'étapes. Nous parlerons assez en détail des liens qui commencent à apparaître entre ces grammaires et la théorie des langages standard, en particulier de la complétude du calcul de Lambek pour ses modèles à base de monoïdes libres [15], et de la résolution par Mati Pentus en 1993 [52] de la conjecture formulée par Chomsky en 1963 [20] : les langages engendrés par les grammaires de Lambek sont algébriques (non contextuels). Hormis les résultats nouveaux pour lesquels on renvoie aux articles originaux, on trouvera davantage de détails dans nos notes de cours [59] ou, sur les questions d'acquisition de grammaire, dans un article de synthèse [7].

2. Grammaires catégorielles

Comme nous l'avons dit dans le premier volet, les grammaires non contextuelles sont une approximation raisonnable de la syntaxe des langues humaines. Nous avons également entrevu qu'il y a un lien étroit entre la structure syntaxique d'une phrase et sa structure logique. Nous allons présenter ici avec un peu plus de précision une famille de formalismes qui décrivent l'analyse syntaxique d'une

phrase comme une déduction dans une logique des ressources (logique linéaire non commutative)[27, 1].

Le principal avantage de ce type de formalisme est la possibilité de calculer la structure logique d'une phrase à partir de sa structure syntaxique. Nous pourrons donc mieux, dans ce genre de formalisme, faire le lien avec des questions de sémantique logique évoquées au paragraphe 1.3 du premier volet, en particulier lorsqu'il s'agit de questions proches de la syntaxe comme la portée des quantificateurs.

Un autre intérêt de ce genre de syntaxe logique est la possibilité d'apprendre ces grammaires à partir d'exemples positifs, comme nous le verrons à la section 5, ce qui est l'un des deux critères de Chomsky pour un formalisme syntaxique comme mentionné au paragraphe 3.2 du premier volet. Quant à l'autre critère, la polynomialité de l'analyse en fonction du nombre de mots, elle résulte de l'équivalence avec les grammaires non contextuelles, ce qui sera un des résultats fondamentaux de la section 6.3 : la conjecture de Chomsky 1963 résolue par Pentus en 1993.

Comme nous l'avons dit plus haut, les catégories syntaxiques ont un pendant sémantique et logique. Un verbe transitif est un prédicat à deux places, et il peut être vu comme une fonction à deux arguments de type individu (groupes nominaux) qui produit une proposition. Ce genre de modèle, qui s'inscrit dans une tradition que l'on peut faire remonter aux recherches logiques d'Edmund Husserl [32, 29, 19] voire à Aristote, a été d'abord introduit pour vérifier la bonne formation des formules logiques par Kazimierz Ajdiukiewicz [3], et se présentait comme un calcul de fractions. Tout simplement, si t et e désignent respectivement la catégorie des valeurs de vérité et la catégorie des individus, un prédicat à un argument est de catégorie $\frac{t}{e}$, un prédicat à deux arguments est de catégorie $\frac{t}{ee}$, une opération logique comme $\wedge$ est de catégorie $\frac{t}{tt}$, et la formule *Dort Pierre* $\wedge$ *Regarde Marie Pierre* est bien formée, de catégorie $t = (\frac{t}{e}e)(\frac{t}{tt})(\frac{t}{ee}ee)$ – on verra ci-après que pour traiter convenablement les quantificateurs, il ne faut pas considérer que $\frac{t}{(\frac{t}{e})}$ soit égal à e.

À la différence des formules logiques en notation polonaise préfixée, les langues naturelles ont un ordre des mots qui leur est propre : en français l'ordre est généralement Sujet Verbe Objet, tandis qu'en japonais c'est plutôt Sujet Objet Verbe et en arabe classique Verbe Sujet Objet. Il faut donc plutôt utiliser des fractions non commutatives, notées $a \setminus b$ et b / a comme l'a proposé Bar-Hillel en 1953 [9], soit à peu près en même temps qu'ont été introduites les grammaires de réécriture usuelles présentées dans le premier volet.

2.1. Catégories, séquents et dérivations

L'idée est d'associer à chaque mot une catégorie (aussi appelée formule, car il s'agit des formules d'une logique propositionnelle) qui exprime les compléments qui lui font défaut, à gauche et à droite. L'ensemble cat des catégories est le plus petit ensemble contenant des catégories de base et clos par les deux opérations $\setminus$ (lu *sous*) et $/$ (lu *sur*). Les catégories de base (aussi appelées variables propositionnelles) contiennent en général S : phrase, *sn*, syntagme (ou groupe) nominal, *n* nom commun, *sv* syntagme (ou groupe) verbal, et éventuellement quelques autres, comme les groupes prépositionnels, les groupes verbaux, etc. On peut écrire cela $\mathrm{cat} ::= \mathrm{P} \quad | \quad \mathrm{cat} \setminus \mathrm{cat} \quad | \quad \mathrm{cat} / \mathrm{cat}$ avec $\mathrm{cat} = \{S, sn, n, \ldots\}$. Les sous catégories

(ou sous formules) d'une catégorie (ou d'une formule) sont les catégories qui figurent à l'intérieur de cette catégorie ; par exemple, $(a \setminus b)$ est une sous catégorie de $d \,/\, ((a \setminus b) \,/\, c)$ mais pas $(d \,/\, a)$.

Ainsi on affectera par exemple la catégorie $(sn \setminus S) \,/\, sn$ à un verbe transitif : celui-ci doit recevoir à sa droite son complément d'objet, puis, à sa gauche, son sujet, et tous deux sont de catégorie sn. On peut rapprocher cette catégorie de l'expression $(sn^{-1}S)sn^{-1}$ mais on se méfiera de cette ressemblance : $(a^{-1}b)^{-1}$ ne se réduit pas à (ba^{-1}) et des expressions comme a^{-1} ou aa ne correspondent pas à des catégories.

Une grammaire catégorielle est définie par la donnée d'une application Lex des mots (ou terminaux) dans les parties finies de cat. Les règles que l'on verra ci-après, dites de réduction ou de déduction, sont universelles : c'est la fonction Lex qui détermine le langage engendré par la grammaire.

Définition 2.1. *Étant donnée une grammaire catégorielle de lexique* Lex*, une suite de mots $m_1 \dots m_n$ est de catégorie U si pour chaque mot m_i il existe un type $t_i \in \mathrm{Lex}(m_i)$ tel que la séquence de catégories $t_1, \dots, t_n$ se réduise en U, ce que l'on notera $t_1, \dots, t_n \vdash U$. Le langage engendré est l'ensemble des suites de mots de catégorie S.*

On observe des différences de conception entre les habituelles grammaires de réécriture :

Terminaux = mots	Terminaux = mots
Non-terminaux sans structure.	Catégories structurées.
Règles de production qui déterminent le langage engendré, écrites indépendamment des non terminaux.	Règles de réduction, indépendantes du langage décrit, qui s'appliquent en fonction de la structure des catégories.

Les règles universelles utilisées sont de deux sortes.

– Les règles d'élimination, les seules considérées par Bar-Hillel affirment concernant $\setminus$ que

- si une suite de mots $m_1 \dots m_n$ est de catégorie a
- et si une suite de mots $w_1 \dots w_p$ est de catégorie $a \setminus b$
- alors la suite de mots $m_1 \dots m_n w_1 \dots w_p$ est de catégorie b

et, concernant $/$ que

- si une suite de mots $m_1 \dots m_n$ est de catégorie $b \,/\, a$
- et si une suite de mots $w_1 \dots w_p$ est de catégorie a
- alors la suite de mots $m_1 \dots m_p \; w_1 \dots w_n$ est de catégorie b

– Les règles d'introduction, que Lambek a apportées en 1958, sont les réciproques des règles d'élimination. Elles font des réductions de catégories un système déductif comme on le verra ci-après, en particulier aux paragraphes 3.3 puis 6.1. Pour $\setminus$, elles disent que :

- si la suite de mots $m_1 \dots m_n w_1 \dots w_p$ est de catégorie b
- et si la suite de mots $m_1 \dots m_n$ est de catégorie a
- alors la suite de mots $w_1 \dots w_p$ est de catégorie $a \setminus b$

tandis que pour $/$ elles affirment symétriquement que :

– si la suite de mots $m_1 \dots m_n w_1 \dots w_p$ est de catégorie b
– et si une suite de mots $w_1 \dots w_p$ est de catégorie a
– alors la suite de mots $m_1 \dots m_n$ est de catégorie $b \,/\, a$

En transformant nos jugements *la suite de mots est de catégorie X* en jugements de la forme *la suite des catégories correspondantes se réduit en X*, on obtient les règles de réduction de la figure 1. Les points de départ des dérivations sont des axiomes $A \vdash A$, où la catégorie A est librement choisie. Les expressions manipulées par ces preuves sont appelés *séquents* et leur symbole principal est $\vdash$: à gauche de ce signe figure une suite finie de catégories appelées *hypothèses du séquent* où les majuscules grecques désignent des suites de catégories, et à droite *une* catégorie appelée *conclusion du séquent*. Ces règles se lisent *si la ou les réductions au dessus de la barre sont déjà établies, alors on a celle inscrite en dessous*. Lorsqu'un séquent s'obtient à partir des axiomes en appliquant les règles il est dit *dérivable* (comme dans un système grammatical) ou *démontrable* (puisqu'il s'agit aussi d'un système déductif). Les dérivations sont aussi appelées *preuves* ou *démonstrations*.

Les grammaires catégorielles de Yehoshua Bar-Hillel [9] sont appelées grammaires AB (pour Ajdukiewicz Bar-Hillel) ou *Basic Categorial Grammars* : elles n'utilisent que l'axiome et les deux règles $/_e$ et $\backslash_e$, parmi les quatre règles de la figure 1. Les grammaires de Lambek utilisent toutes les règles.

$$\frac{}{A \vdash A}\ \textit{axiome}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash A \setminus B}{\Gamma, \Delta \vdash B}\setminus_e \qquad \frac{A, \Gamma \vdash C}{\Gamma \vdash A \setminus C}\setminus_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Delta \vdash B \,/\, A \quad \Gamma \vdash A}{\Delta, \Gamma \vdash B}/_e \qquad \frac{\Gamma, A \vdash C}{\Gamma \vdash C \,/\, A}/_i \quad \Gamma \neq \varepsilon$$

FIG. 1. Les règles des grammaires AB (les deux règles de gauche, $/_e, \backslash_e$) et celles du calcul de Lambek en déduction naturelle (les quatre règles $/_e, \backslash_e, /_i, \backslash_i$). Les majuscules grecques désignent des suites de formules.

3. Exemples de dérivations et structure des analyses

3.1. Grammaires AB

Donnons un petit exemple de lexique/grammaire emprunté à une langue cousine du français, l'italien qui un peu plus loin, aura pour nous l'avantage de ne pas avoir d'inversion du sujet dans les questions.

Mot	Catégorie(s)	Traduction
cosa	$(S \,/\, (S \,/\, sn))$	*que* (interrogatif)
guarda	$(S \,/\, sv)$	(il, elle) *regarde*
passare	$(sv \,/\, sn)$	*passer*
il	$(sn \,/\, n)$	*le* (article)
treno	n	*train*

La phrase *guarda passare il treno* (*il regarde passer le train*) appartient au langage engendré, puisque le séquent $(S/sv), (sv/sn), (sn/n), n \vdash S$ est dérivable.

On peut comme dans les règles de la figure 2 écrire tous les contextes, ce qui donne :

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

Il existe plusieurs manières de représenter un arbre de dérivation et on peut aussi, sans perte d'information, ne noter que les conclusions des séquents, et dans ce cas les feuilles forment la suite des catégories qui correspondent aux mots.

$$\dfrac{(S \,/\, sv) \qquad \dfrac{(sv \,/\, sn) \qquad \dfrac{(sn \,/\, n) \quad n}{sn}\,/_e}{sv}\,/_e}{S}\,/_e$$

L'arbre de dérivation peut aussi être représenté ainsi :

$$[/_e(S \,/\, sv) \quad [/_e(sv \,/\, sn) \quad [/_e(sn \,/\, n) \quad n]]]$$

Considérons maintenant la phrase *Cosa guarda passare*? (Que regarde-t-il passer ?). Elle n'appartient pas au langage engendré puisque la suite de catégories

$$(S \,/\, (S \,/\, sn))(S \,/\, sv)(sv \,/\, sn)$$

ne contient jamais deux catégories successives qui se réduisent.

3.2. Grammaires de Lambek

Les grammaires de Lambek sont définies comme les grammaires catégorielles de base – en particulier ce sont toujours des grammaires lexicalisées – à ceci près que l'on peut introduire des hypothèses, c'est-à-dire faire comme s'il y avait une suite de mots d'une catégorie donnée, puis supprimer cette hypothèse par les nouvelles règles d'introduction. Avec le même lexique que celui donné pour la grammaire AB ci-dessus, nous donnons en figure 2 la dérivation de $(S/(S/np)), S/vp, vp/np \vdash S$ qui montre que *cosa guarda passare* est une phrase – au passage, on dérive la transitivité de /.

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

FIG. 2. Une déduction naturelle dans le calcul de Lambek

On peut se passer des contextes, ce qui donne un arbre plus aisément lisible, comme celui ci-dessous. Le contexte associé à une formule F est l'ensemble des feuilles, qui ne sont pas entre crochets ou dont les crochets qui les entourent sont associées à une règle d'introduction située plus bas que F(on verra ci-après qu'on peut reconstruire cette information).

$$\dfrac{(S \,/\, (S \,/\, sn)) \qquad \dfrac{\dfrac{(S \,/\, sv) \qquad \dfrac{(sv \,/\, sn) \quad [sn]_1}{sv}\,/_e}{S}\,/_e}{S \,/\, sn}\,/_i(1)}{S}\,/_e$$

3.3. Structure des dérivations dans les grammaires de Lambek

Nous proposons deux formulations du calcul de Lambek, et nous les comparons ici brièvement, ce sont du reste des résultats assez immédiats, qui s'obtiennent par récurrence sur la hauteur des preuves. La première propriété est une bonne nouvelle : les mêmes séquents sont dérivables dans l'un et l'autre système et la correspondance est relativement facile à établir.

$$\dfrac{}{A \vdash A}\ axiome \qquad \dfrac{\Gamma \vdash K \qquad \Delta, K, \Theta \vdash C}{\Delta, \Gamma, \Theta \vdash C}$$

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

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

FIG. 3. Les règles du calcul de Lambek, calcul des séquents. Les majuscules grecques désignent des suites de formules qui peuvent être vides.

Le lecteur est en droit de se demander pourquoi donner deux ensembles similaires de règles indigestes. En fait, il est difficile de se passer d'un de ces deux calculs :

le premier est plus adapté aux calcul de représentations sémantiques et le second est mieux adapté à la correspondance avec la théorie des langages standard.

La première remarque qui s'impose, c'est qu'en lisant $A \setminus B$ et B / A comme $A \Rightarrow B$, les règles du calcul de Lambek sont les règles usuelles de l'implication, ou plus précisément une restriction des règles usuelles, puisqu'on ne peut ni ajouter d'hypothèses, ni en fusionner (linéarité), ni même en permuter (non commutativité) : nous avons une authentique suite de formules sans à gauche de $\vdash$.

Une autre remarque concerne les deux formulations : dans un séquent $\Gamma \vdash X$ la suite d'hypothèses Γ n'est jamais vide (elle ne l'est pas initialement, et ne peut jamais le devenir à cause de la restriction sur les règles $\setminus_i$ et $/_i$), ce qui peut étonner. Du point de vue des suites de mots, cela revient à ne jamais affecter de catégorie à la suite vide. Si on omettait cette restriction, la suite vide ε aurait effectivement des catégories, toutes les catégories A telles que $\vdash A$, et notamment n / n (la catégorie des adjectifs antéposés). On pourrait alors donner la suite vide comme argument à une suite de mots ou un mot de catégorie de la forme $A \setminus B$ ou A / B, et par exemple à *très* qui est de catégorie $(n / n) / (n / n)$ (la catégorie des modifieurs d'adjectifs antéposés). Et tandis qu'on pensait qu'une suite de mots de catégorie A / B (ici $(n / n) / (n / n)$) *nécessitait* à sa droite une suite de mots de catégorie A (ici n / n) pour donner une suite de mots de catégorie B, avec la séquence vide, on pourrait montrer que *un très exemple* est un groupe nominal (tout comme si le vide entre *très* et *exemple* était un adjectif antéposé de catégorie n / n, comme le serait l'adjectif *simple*).

$$\dfrac{\text{un} : np / n \qquad \dfrac{\text{très} : (n / n) / (n / n) \qquad \dfrac{[n]_\alpha}{n / n} /_i -\alpha}{n / n} /_e \qquad \text{exemple} : n}{np} /_e \qquad\qquad n /_e$$

Dans le calcul des séquents, les règles $\setminus_e$ et $/_e$ sont valides, mais on leur préfère des règles $\setminus_h$ et $/_h$ d'introduction de connecteurs en hypothèses, du côté gauche de $\vdash$. Voici de nouveau la dérivation de $(S / (S / np)), S / vp, vp / np \vdash S$ qui montre que *cosa guarda passare* est une phrase mais dans le calcul des séquents cette fois.

$$\dfrac{\dfrac{S \vdash S \quad vp \vdash vp}{S / vp, vp \vdash S} /_h \quad np \vdash np}{S / vp, vp / np, np \vdash S} /_h \qquad \dfrac{S \vdash S \qquad S / vp, vp / np \vdash S / np}{(S / (S / np)), S / vp, vp / np \vdash S} /_i \; /_h$$

Mentionnons brièvement quelques propriétés du calcul des séquents. L'axiome peut-être restreint aux catégories élémentaires puisque $A \vdash A$ est dérivable à partir de $p \vdash p$ et des autres règles (sans la coupure). La règle de coupure est redondante : toute preuve d'un séquent peut être transformée en une preuve sans coupure aussi dite *normale* (ce résultat s'établit facilement pour ce calcul, mais pour les logiques usuelles comme la logique classique, c'est plus délicat). On remarquera que les

dérivations sans coupure satisfont la propriété de la sous formule : toute formule apparaissant dans l'un des séquents d'une dérivation est une sous formule d'une formule du séquent établi.

Dans la déduction naturelle une dérivation est dite *normale* lorsqu'on n'introduit jamais une formule $A \setminus B$ par la règle $\setminus_i$ pour immédiatement l'éliminer par la règle $\setminus_e$ (et bien sûr on traite $/$ de la même manière). Il est aussi possible de transformer toute dérivation en une dérivation normale du même séquent, la transformation étant légèrement différente de l'élimination des coupures. Une propriété extrêmement importante et particulière au calcul de Lambek, si on le compare à d'autres formalismes logiques, logique intuitionniste ou même linéaire est la suivante : il suffit de noter les règles utilisées et les catégories syntaxiques de départ (celles des axiomes) pour connaître la preuve, car l'hypothèse A supprimée par la règle $\setminus_i$ ou $/_i$ peut-être déterminée à partir de l'arbre en partant des feuilles vers la racine, car c'est toujours l'hypothèse la plus à droite pour $/_i$ ou la plus à gauche pour $\setminus_i$ qui est supprimée. Une dérivation est donc un véritable arbre au sens usuel, dont les noeuds internes sont soit binaires et étiquetés par $\setminus_e$ ou par $/_e$, soit unaires et étiquetés par $\setminus_i$ ou par $/_i$ tandis que les feuilles sont des catégories – et pour les dérivations normales, on peut supposer que ces catégories sont des sous catégories de celles qui figurent dans le lexique. Ainsi on peut noter, sans perte d'information, notre déduction naturelle de la figure 2 ainsi :

$$/_e[(S \,/\, (S \,/\, sn)), /_i[/_e[S \,/\, sv, /_e[sv \,/\, sn, sn]]]]$$

Dans le calcul des séquents, comme dans la déduction naturelle, la possibilité de transformer les dérivations en dérivations du même séquent satisfaisant la propriété de la sous formule garantit qu'un séquent est dérivable si et seulement s'il l'est au moyen d'une preuve ne contenant que des sous formules du séquent qu'on souhaite dériver. On en déduit relativement aisément qu'on peut décider en un nombre fini d'opérations si un séquent est ou non démontrable dans le calcul de Lambek. Les formules apparaissant dans toute analyse syntaxique sont de taille inférieure ou égales à la taille maximale d'une formule du lexique – en revanche, le nombre de formules en hypothèse d'un séquent, c'est-à-dire la longueur de la suite de mots analysées n'est pas bornée[39].

Théorème 3.1 (Lambek)**.** *Toute dérivation dans le calcul de Lambek admet une forme normale qui satisfait la propriété de la sous formule. Par conséquent, le calcul de Lambek est décidable ainsi que l'appartenance d'une phrase au langage engendré par une grammaire de Lambek.*

Les deux résultats de complexité qui suivent closent deux questions longtemps ouvertes. Ils sont tous les deux dus à Mati Pentus [52, 53] et le premier fera l'objet de la section 6.3 – pour être précis, notre formulation du premier est un corollaire dû à Alain Finkel et Isabelle Tellier [25].

Théorème 3.2 (Pentus)**.** *L'analyse d'une phrase dans le calcul de Lambek est un problème polynomial (cubique) en fonction du nombre de mots de la phrase. La prouvabilité d'un séquent dans le calcul de Lambek est un problème NP complet (en fonction de la taille du séquent).*[4]

[4] Ma formulation du second résultat n'est pas tout à fait exacte car son résultat porte sur le calcul de Lambek *avec produit*. J'ai ici ignoré ce connecteur, qui *grosso modo* correspond à la

Si on pense, comme on le fait dans le calcul de Lambek, qu'une analyse syntaxique est une preuve alors la syntaxe ordinaire de la logique est bien peu satisfaisante. En effet, des preuves peuvent ne différer que pour des raisons inessentielles comme l'ordre dans lequel les règles sont appliquées. Il est possible et souhaitable de quotienter les preuves par les permutations de règles. Ce travail a tout d'abord été fait pour la logique linéaire de Jean-Yves Girard, dans l'article originel [27] qui a introduit la notion de *réseau de démonstration* : ce sont des graphes qui sont le quotient des preuves usuelles par les permutations de règles. Ils permettent d'avoir à la fois une définition inductive des preuves et aussi une caractérisation universelle des preuves comme l'ensemble des graphes satisfaisant une propriété universelle (de la forme $\forall x \exists y \cdots$).

Il existe plusieurs variantes équivalentes des réseaux de démonstrations qui quotientent les preuves par les permutations de règles. Néanmoins, seule la nôtre [58] permet aussi de quotienter les formules par les propriétés algébriques des connecteurs (associativité et commutativité le cas échéant). Initialement conçue pour le calcul commutatif, nous l'avons récemment étendue au calcul non commutatif avec Sylvain Pogodalla [56]. C'est cette dernière que nous allons ici présenter.

L'expression « l'associativité des connecteurs » peut surprendre puisque $(A \setminus B) \setminus C \not\equiv A \setminus (B \setminus C)$.... L'associativité des connecteurs est celle de connecteurs que nous avons volontairement laissés implicites. Ce sont ceux de la logique linéaire, logique qui dispose d'une disjonction, d'une conjonction et d'une négation. La conjonction correspond à la virgule : on aurait pu écrire un séquent $A, B, C \vdash X$ comme $A \wedge B \wedge C \vdash X$. Cette virgule est bien associative et, comme on l'a fait remarquer, non commutative. La disjonction permet avec la négation notée d'exprimer l'implication : $A \setminus B \equiv \neg A \vee B$ et (ce qui est l'analogue de $(A \Rightarrow B) \equiv (\neg A \vee B)$). Elle n'est pas commutative, et c'est pour cela que nous avons deux implications, l'autre étant $B \,/\, A \equiv B \vee \neg A$. L'associativité de ce connecteur est perceptible dans le calcul de Lambek par le fait que l'on a bien $(A \setminus X) \,/\, B \equiv A \setminus (X \,/\, B)$ qui s'écrit aussi : $(\neg A \vee X) \vee \neg B \equiv \neg A \vee (X \vee \neg B)$ ce qui est bien de l'associativité.

Voyons comment associer à une catégorie syntaxique C une formule de la logique linéaire $\ell(C)$ dont nous noterons les opérations $\wedge$ (conjonction) $\vee$ (disjonction) $\neg$ (négation) – en logique linéaire la conjonction s'écrit habituellement $\otimes$, la disjonction comme un & à l'envers et la négation $(\cdots)^{\perp}$.

- $\ell(p) = p$
- $\ell(F \setminus G) = \neg\ell(F) \vee \ell(G)$
- $\ell(G \,/\, F) = \ell(G) \vee \neg\ell(F)$

La logique linéaire non commutative vérifie les lois d'Augustus de Morgan qui s'écrivent $\neg\neg X = X$, $\neg(A \wedge B) = \neg B \vee \neg A$, $\neg(A \vee B) = \neg B \wedge \neg A$, non commutativité oblige. En utilisant ces lois, il est possible d'écrire toute formule de la logique linéaire et donc tout $\ell(C)$ ou même $\neg\ell(C)$ comme une formule de la logique linéaire, avec des $\wedge$ des $\vee$ et une ou zéro négation sur chaque catégorie de base : c'est ce qu'on appelle la *forme normale négative d'une formule*. Les occurrences

virgule à gauche : $A, B \vdash C$ et $A \bullet B \vdash C$ sont équiprouvables, $(A \setminus (B \setminus C)$ et $(B \bullet A) \setminus C$ sont équivalents, etc. Du calcul de Lambek avec $\setminus$ et $/$ nous ne savons pour le moment qu'une chose : il est dans NP, mais je serais surpris qu'il y ait une grande différence de complexité entre les calculs de Lambek avec et sans produit.

des catégories de bases sont appelées atomes. Lorsqu'elles sont non niées on parle d'atomes positifs et lorsqu'elles le sont (et alors elles le sont exactement une fois, puisque $\neg\neg p = p$) on parle d'atomes négatifs.

Mot m	Catégorie $c(m)$	$\ell(c(m))$	$\neg\ell(c(m))$	Forme Normale Négative
cosa	$(S \,/\, (S \,/\, sn))$	$(S \vee \neg(S \vee \neg sn))$	$\neg(S \vee \neg(S \vee \neg sn))$	$(S \vee \neg sn) \wedge \neg S$
guarda	$(S \,/\, sv)$	$(S \vee \neg sv)$	$\neg(S \vee \neg sv)$	$sv \wedge \neg S$
passare	$(sv \,/\, sn)$	$(sv \vee \neg sn)$	$\neg(sv \vee \neg sn)$	$sn \wedge \neg sv$

Si on se penche sur le calcul de Lambek, tel qu'il est formulé dans le calcul des séquents, on peut se demander ce qu'il faut connaître pour reconstituer une preuve. Une fois qu'on a remarqué que les catégories de base sont introduites deux par deux dans un axiome $p \vdash p$, une positive et une négative, qu'on peut les suivre tout au long d'une preuve normale, on peut deviner qu'il suffit de connaître les paires de a introduites par un même axiome. C'est effectivement suffisant pour une preuve normale, sans coupure.

Étant donné un couple d'atomes (x, y) d'une formule F on dit qu'il se rencontre sur l'occurrence $*$ d'un connecteur s'il existe une sous formule $(G * H)$ de F telle que x est dans G et y dans H.

Étant donnée une formule de la logique linéaire, on lui associe un graphe orienté dont les arêtes sont $\mathfrak{r}$ (rouge) ou $\mathfrak{n}$ (noir) ainsi : les sommets du graphe sont les atomes de la formule, et on place un arc $\mathfrak{n}$ de x vers y si le couple (x, y) se rencontre sur une disjonction, et un arc $\mathfrak{r}$ de x vers y s'ils se rencontrent sur une conjonction. On remarquera que la relation *il existe un arc* $\mathfrak{r}$ *ou* $\mathfrak{n}$ *de* x *vers* y est un ordre total, que le graphe $\mathfrak{r}$, tout comme le graphe $\mathfrak{n}$ est un ordre serie-parallèle et que le graphe non orienté sous-jacent est un graphe complet. Sur les ordres série-parallèle, maintes fois redécouverts, on pourra consulter [46]

Rappelons (voir par exemple [45]) qu'un *ordre cyclique total* est une relation ternaire $C(x, y, z)$ telle que $C(x, y, z) \Rightarrow C(y, z, x)$ (cyclicité), $C(x, y, z) \wedge C(x, u, y) \Rightarrow C(u, y, z)$ (transitivité) et $C(x, y, z) \vee C(x, z, y)$ (totalité). La relation $C(x, y, z)$ indique comment les points sont disposés sur un cercle orienté : en allant de x à z dans le sens positif on rencontre y. Une application bijective σ telle que $\sigma(x) \neq x$ et $\sigma(\sigma(x)) = x$ pour tout x est dite *compatible* avec C si et seulement si $\forall x, u \quad C(x, u, \sigma(x)) \Rightarrow C(x, \sigma(u), \sigma(x))$

Étant donné un séquent $\mathbb{S} = A_1 \ldots, A_n \vdash C$ on lui associe un graphe $\mathfrak{r}$ et $\mathfrak{n}$ comme suit. Dans un premier temps, on construit les graphes $\mathfrak{r}$ et $\mathfrak{n}$ des formes normales négatives de $\ell(C)$, $\neg\ell(A_1)$, $\ldots$, $\ell\neg(A_n)$. On ajoute ensuite des arcs $\mathfrak{n}$ pour ordonner cycliquement les formules : un arc $\mathfrak{n}$ du dernier atome (qui est le maximum du point de vue de l'ordre total) de $\neg\ell(A_i)$ vers le premier (le minimum du point de vue de l'ordre) de $\neg\ell(A_{i-1})$, du dernier atome de $\ell(C)$ vers le premier de $\neg\ell(A_n)$ et du dernier de $\neg\ell(A_1)$ vers le premier de $\ell(C)$. Ce graphe contient un cycle hamiltonien orienté, qui donne un ordre cyclique aux atomes. Il est aisé de reconnaître si un graphe bicolore est la traduction graphique d'un séquent du calcul de Lambek.

On peut alors donner une caractérisation des dérivations avec des termes standard de la théorie des graphes, dont le principal intérêt est de donner aux dérivations, en plus de la définition inductive usuelle (issue de la notion de dérivation vue ci-dessus), une définition comme l'ensemble des structures (des graphes colorés) satisfaisant certaines propriétés universelles (de la forme $\forall x \exists y \cdots$). C'est un premier pas vers une caractérisation du genre modèles finis ou langage de graphes.

Pour traduire une dérivation en un graphe, on peut procéder inductivement, mais en fait il suffit de connaître les atomes provenant d'un même axiome $p \vdash p$: il suffit de construire le graphe $\mathfrak{r}$ et $\mathfrak{n}$ du séquent, et d'ajouter une arête non orientée d'une troisième couleur, $\mathfrak{b}$ (bleu) entre les atomes provenant d'un même axiome.

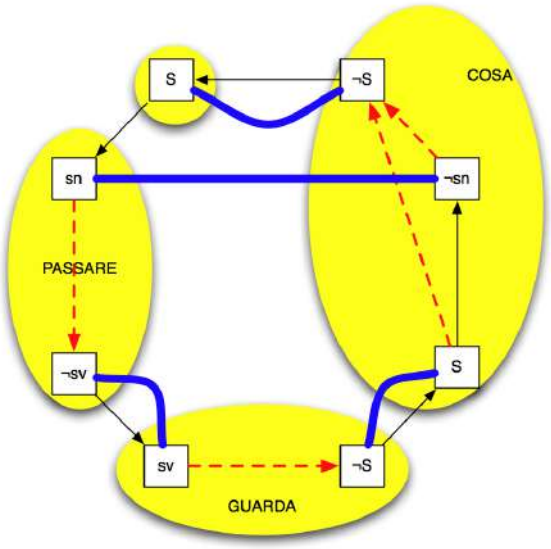

FIG. 4. Un réseau de démonstration, le graphe d'analyse de la phrase *cosa guarda passare*. Les arêtes $\mathfrak{b}$ sont les plus épaisses (non orientées) les $\mathfrak{r}$ sont en pointillés, et les arêtes $\mathfrak{n}$ fines et pleines. Les ellipses indiquent les atomes appartenant à une même catégorie.

Théorème 3.3 (Pogodalla, Retoré)**.** *Un graphe dont les sommets sont des atomes et dont les arêtes sont $\mathfrak{r}$ (orientées), $\mathfrak{n}$ (orientées) ou $\mathfrak{b}$ (non orientées) correspond à au moins une démonstration du calcul de Lambeck si et seulement si :*

– Les arêtes $\mathfrak{b}$ joignent des occurrences de polarité opposée d'une même catégorie de base. Elles ne sont jamais adjacentes et couvrent tous les sommets du graphe (elles forment un couplage parfait).

– La partie $\mathfrak{r}$ et $\mathfrak{n}$, orientée, décrit des formules du calcul de Lambek et un ordre cyclique entre elles, au sens où on l'a défini ci-dessus, et contient un cycle hamiltonien orienté.

– *Le graphe non orienté* $\mathfrak{r}$ *et* $\mathfrak{b}$ *obtenu en supprimant les arêtes* $\mathfrak{n}$ *et en omettant l'orientation des arêtes rouges, a la propriété suivante : tout cycle élémentaire alternant contient une corde.*[5]

– *La fonction* σ *qui à un sommet associe l'autre extrémité de l'unique arête* $\mathfrak{b}$ *qui lui est adjacente est compatible avec l'ordre cyclique défini par le circuit hamiltonien du séquent.*

On trouvera sur la figure 4 notre analyse dans le calcul de Lambek de *cosa guarda passare* vue comme un réseau de démonstration. Ce genre de représentation capte mieux l'essence de la preuve, et si on pense qu'une preuve est une analyse syntaxique, alors il ne faut qu'une analyse par preuve, alors que plusieurs preuves du calcul des séquents peuvent correspondre à une même analyse. De plus ce genre de représentation permet d'utiliser des algorithmes sur les graphes qui sont plus efficaces pour construire des preuves ou analyses syntaxiques, comme par exemple dans [49] .

4. De la structure syntaxique d'une phrase à sa structure logique

Nous avons affirmé voire répété que la syntaxe catégorielle permettait de convertir une analyse syntaxique en une formule logique. Nous expliquons dans cette section comment cela fonctionne. Tout d'abord, il faut représenter la logique d'ordre supérieur dans le λ-calcul typé, car il permet de gérer proprement la composition, pour appliquer le principe frégéen suivant lequel le sens du tout est fonction du sens des parties. Ceci a été développé par Richard Montague au début des années soixante-dix en combinant la représentation de la logique d'ordre supérieur en λ-calcul typé d'Alonso Church, ses propres idées sur la logique intentionnelle et son interprétation dans des mondes possibles à la Kripke, voir [26, 65], mais nous ne parlerons pas ici de ces deux derniers points. Nous nous contenterons de montrer comment une analyse dans le calcul de Lambek permet de calculer automatiquement les formules logiques associées à la phrase analysée. Cette correspondance est bien entendu liée à la nature logique des analyses dans le calcul de Lambek.

4.1. Des catégories syntaxiques aux types sémantiques

La description de la logique dans le λ-calcul typé, fondé sur la notion de fonction, représente les ensembles au moyen de leurs fonctions caractéristiques. Par exemple, une relation n-aire est une fonction qui associe à un n-uplet la valeur VRAI s'il est dans la relation et la valeur FAUX sinon – on peut éventuellement utiliser d'autres valeurs de vérités. Les types de base sont les individus ou entités e ainsi que les valeurs de vérités t, et l'ensemble des types est le plus petit qui contienne e, t et soit clos par l'opération $\rightarrow$:

$$types ::= e \quad | \quad t \quad | \quad types \rightarrow types$$

Le type $A \rightarrow B$ est bien évidemment celui des fonctions de A dans B. Le produit n'est pas nécessaire : une fonction à deux arguments de $A \times B$ dans C peut se voir comme une fonction de A dans $B \rightarrow C$, l'ensemble des

[5] C'est la seule condition dans le cas commutatif, avec une autre condition de connexité, mais elle est ici automatiquement assurée parce que le graphe $\mathfrak{r}$ et $\mathfrak{n}$ provient de catégories syntaxiques du calcul de Lambek et non de formules quelconques de la logique linéaire.

fonctions de B dans C, Plus généralement, une fonction $A_1 \times \cdots \times A_n$ dans C sera vue comme un élément de $(A_1 \to (A_2 \to (A_3 \to (\cdots (A_{n-1} \to (A_n \to C))))))$.

Un nom commun tel *chaise* aura pour type sémantique $e \to t$ et sera interprété comme une fonction qui s'applique à des entités et vaudra VRAI des entités qui sont des chaises. Un prédicat unaire comme *dort* sera interprété de même. Un verbe transitif comme *regarde* aura pour type sémantique $e \to (e \to t)$ et sera interprété comme une fonction qui vaut VRAI lorsqu'on l'applique successivement à deux entités x et y dont la première regarde la seconde. La parenté avec les catégories syntaxique est manifeste, et plus précisément on peut définir un morphisme $(\_)^*$ des catégories syntaxiques vers les types sémantiques qui matérialise les intuitions logico-mathématiques helléniques :

(Catégorie syntaxique)*	=	Type Sémantique type	
S^*	=	t	une phrase est une proposition
sn^*	=	e	un syntagme nominal est une entité
n^*	=	$e \to t$	un nom commun est un sous-ensemble des entités, un prédicat à une place
$(a \setminus b)^* = (b \,/\, a)^*$	=	$a^* \to b^*$	étend $(\_)^*$ à toutes les catégories syntaxique

Le lexique qui jusqu'ici associait à chaque mot une catégorie syntaxique u, va maintenant aussi lui associer un λ-terme dont le type est u^*. Un *λ-terme* est une expression du λ-calcul, ensemble de termes et système de réécriture fondé sur la notion de fonction, qui se distingue ainsi des visions ensemblistes. L'ensemble des λ-termes simplement typés est défini ainsi :

– Chaque type a contient les termes suivants : des variables de type a qui sont libres dans le terme qu'elles constituent, et éventuellement des constantes de type a.

– Si x une variable de type a et si u est un terme de type b alors $\lambda x.u$ est un terme de type $a \to b$. C'est la fonction qui à x associe u, terme qui généralement contient x. Les variables libres de $\lambda x.u$ sont celles de u exceptée x (toutes les occurrences libres de x dans u sont devenues liées).

– Si u est un terme de type $a \to b$ et v est un terme de type a, alors $(u(v))$ est un terme de type b. On applique la fonction de a dans b à un élément de a. Les variables libres de $(u(v))$ sont celles de u et celles de v.

L'expression $t : A$ signifie que λ-terme t est de type A. Un vecteur $\vec{X}$ désigne une suite $X_1, \ldots, X_n$ de types, un vecteur $\vec{x}$ désigne une suite $x_1, \ldots, x_n$ de variables et $\vec{x}{:}\vec{X}^*$ signifie $x_1{:}X_1^*, \ldots, x_n{:}X_n^*$ où les x_i sont des variables. Une expression $\vec{x}{:}\vec{X}^* \vdash t : T$ signifie que t est un terme de type T dont les variables libres sont parmi les x_i qui sont respectivement de type X_i.

Le λ-calcul a été initialement inventé pour gérer proprement la substitution dans les tentatives de formalisation totale des mathématiques au début du XX$^{\text{e}}$siècle, et c'est effectivement de cette manière – et non dans la correspondance entre λ-termes et déductions – qu'on va tout d'abord l'utiliser pour modéliser la logique d'ordre supérieur. Le λ-calcul contient une opération

appelée β-réduction qui effectue les substitutions : chaque fois qu'une expression $(\lambda x.u)v$ apparaît dans un terme on peut la β-réduire en u dans lequel la variable x est remplacée par le terme v (on remarquera que par construction x et v ont nécessairement le même type). Pour prendre un exemple simple avec le type de base $\mathbb{N}$ et les constantes $+$ et $*$ de type $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$, que vaut l'expression $((\lambda x \lambda y((+x)((*2)y))((\lambda z.((+z)3))4))((+5)7))$? On pourrait l'écrire $g(h(4), 5+7)$ avec $h : z \mapsto z+3$ et $g : (x,y) \mapsto x+2y$. Elle se réécrit par β-réduction en $((\lambda y.((+((\lambda z.((+z)3))4))((*2)y)))((+5)7))$ puis en $((+((\lambda z.((+z)3))4))((*2)((+5)7)))$ et finalement en $(+((+3)4)((*2)((+5)7)))$ qui ne se réduit plus (et qu'en notation standard, non préfixée on écrirait $(3+4)+(2*(5+7))$.

Pour décrire le langage de la logique, plutôt que des fonctions sur les entiers, nous aurons besoin des connecteurs et quantificateurs. Les connecteurs binaires prennent pour arguments deux propositions et en fabriquent une nouvelle, ils sont donc de type $t \to (t \to t)$ La quantification du premier ordre prend comme argument une fonction des entités dans les propositions et construit une proposition, elle est donc de type $(e \to t) \to t$. En effet, étant donnée une formule à une variable libre, par exemple $\mathit{regarde}(x, \mathit{Marie})$ (que l'on écrit $\lambda x.((\mathit{regarde}\ x)\mathit{Marie})$) pour écrire comme un λ-terme la formule $\forall x\ \mathit{regarde}\ (x, \mathit{Marie})$, on commence par construire la fonction $x \mapsto \mathit{regarde}(x, \mathit{Marie})$ (que l'on écrit $\lambda x.((\mathit{regarde}\ x)\mathit{Marie})$), qui est bien de type $e \to t$ et on lui applique la constante $\forall$ qui fabrique la proposition $\forall x\ \mathit{regarde}\ (x, \mathit{Marie})$ de type t que l'on écrit $\forall(\lambda x.((\mathit{regarde}\ x)\mathit{Marie}))$. On parle de logique d'ordre supérieur car on a naturellement, par exemple, des prédicats du second ordre comme $(e \to t) \to t$ et qu'on peut très bien quantifier sur de tels objets avec un quantificateur de type $((e \to t) \to t) \to t$.

Voici les constantes logiques dont nous aurons besoin pour décrire la logique du premier ordre :

Constante	Type
$\exists$	$(e \to t) \to t$
$\forall$	$(e \to t) \to t$
$\wedge$	$t \to (t \to t)$
$\vee$	$t \to (t \to t)$
$\Rightarrow$	$t \to (t \to t)$

Il nous faudra aussi des constantes pour décrire les termes du lexique :

aime	$\lambda x \lambda y$ (aime x) y	$x : e,\ y : e, \text{aime} : e \to (e \to t)$
aime est un prédicat à deux places		
Pierre	λP (P Pierre)	$P : e \to t,$ Pierre $: e$
Pierre est vu à la Leibnitz comme l'ensemble des propriétés vraies de Pierre		

4.2. Des analyses aux λ-termes

Voyons comment on obtient une représentation sémantique d'une phrase $m_1 \dots m_n$ à partir des ingrédients suivants :

– *une analyse syntaxique, dans le calcul de Lambek en déduction naturelle de* $m_1 \dots m_n$, c'est-à-dire une preuve formelle $\mathcal{D}$ de $t_1, \dots, t_m \vdash S$ et

– *un* λ*-terme sémantique de chaque mot* $m_1, \dots$ *et* m_n, c'est-à-dire des λ-termes τ_i de type t_i^*,

Il suffit de suivre la procédure que nous donnons ici, dont le fonctionnement se comprend aisément, surtout sur un exemple. Nous donnons aussi au passage les raisons de son bon fonctionnement, même si elles nécessitent quelques notions de théorie des types comme [28, 37] que nous ne donnons pas ici (la correspondance de Curry-Howard entre preuves intuitionnistes et λ-termes, le plongement du calcul de Lambek dans la logique intuitionniste).

(1) On remplace toute catégorie syntaxique A de $\mathcal{D}$ par le type sémantique qui lui correspond, A^* ; comme la logique intuitionniste est une extension du calcul de Lambek, on obtient une preuve en logique intuitionniste $\mathcal{D}^*$ de $t_1^*, \dots, t_n^* \vdash t = S^*$.

(2) Par le célèbre isomorphisme de Curry-Howard [31] cette preuve peut se voir comme un λ-terme $\mathcal{D}_\lambda^*$ de type $t^* = S$ qui contient une variable libre x_i de type t_i^* pour chaque mot m_i. Le modus ponens correspond à l'application d'une fonction à un argument et l'introduction correspond à la λ-abstraction, c'est-à-dire à la définition d'une fonction à partir d'un terme ayant une variable libre. Cette correspondance entre règles de construction de l'analyse syntaxique et règles de construction de la représentation sémantique est donnée dans le tableau ci-dessous :

règles syntaxiques	règles sémantiques construction du λ terme
$\dfrac{\vec{X} \vdash A \quad \vec{Y} \vdash A \setminus B}{\vec{X}, \vec{Y} \vdash B} \setminus_e \qquad \dfrac{\vec{Y} \vdash B \,/\, A \quad \vec{X} \vdash A}{\vec{Y}, \vec{X} \vdash B} /_e$	$\dfrac{\vec{x}{:}\vec{X}^* \vdash u{:}A^* \quad \vec{y}{:}Y^* \vdash f{:}A^* \to B^*}{\vec{x}{:}\vec{X}^*, \vec{y}{:}Y^* \vdash (fu){:}B^*} \setminus_e$
$\dfrac{A, \vec{X} \vdash C}{\vec{X} \vdash A \setminus C} \setminus_i \qquad \dfrac{\vec{X}, A \vdash C}{\vec{X} \vdash C \,/\, A} /_i$	$\dfrac{x{:}A^*, \vec{x}{:}\vec{X}^* \vdash t{:}C^*}{\vec{x}{:}\vec{X}^* \vdash (\lambda x.t){:}A^* \to C^*} \setminus_i$
$\dfrac{}{A \vdash A}$ *axiome*	$\dfrac{}{x{:}A^* \vdash x{:}A^*}$ *axiome*

(3) On remplace dans $\mathcal{D}_\lambda^*$. chaque variable x_i par le λ-terme τ_i – dont le type est aussi t_i^*, il s'agit donc d'une substitution correcte.

(4) On réduit le λ-terme de type t obtenu : il s'agit d'un λ-terme sans variable libre et de type t et donc, par suite de résultats assez simples, d'une proposition : c'est la formule logique associée à la phrase.

4.3. Un exemple

On donne ci-dessous un lexique analysant la (!) phrase : « Certains énoncés parlent d'eux-mêmes. » [6]

[6] Toujours au sens de : « *certaines phrases sont circulaires* », comme *Cette phrase est vraie*, ou pire, *Je mens.*

mot	Type syntaxique u Type sémantique u^* Représentation sémantique : λ-terme de type u^* x^v variable ou constante x de type v	
énoncés	$n = E$	*« énoncés » est un nom commun*
	$e \to t = E^*$	*du point de vue sémantique c'est un prédicat à une place*
	$\lambda x^e(\texttt{enonce}^{e\to t}\ x)$	*ce prédicat est la fonction qui a tout individu x associe la valeur de vérité de « x est un énoncé »*
parlent_de	$(sn \setminus S) / sn = P$	*parler_de attend à sa droite un groupe nominal, et à sa gauche un groupe nominal pour produire une phrase*
	$e \to (e \to t) = P^*$	*du point de vue sémantique « parler_de » est un prédicat à deux places*
	$\lambda y^e\ \lambda x^e\ ((\texttt{parler\_de}^{e\to(e\to t)}\ x)y)$	*c'est-à-dire une fonction qui prend deux individus et rend* VRAI *si et seulement si le second argument (le sujet) parle du premier (l'objet)*
eux-mêmes	$((sn \setminus S) / sn) \setminus (sn \setminus S) = X$	*« eux-mêmes » (objet) attend à sa gauche un verbe transitif, pour produire une phrase à la quelle il manque un sujet, c'est-à-dire un verbe transitif*
	$(e \to (e \to t)) \to (e \to t) = X^*$	*du point de vue sémantique, « eux-mêmes » prend un prédicat à deux places $P(x, y)$ (le verbe transitif) et rend un prédicat à une place*
	$\lambda P^{e\to(e\to t)}\ \lambda x^e\ ((P\ x)x)$	*le prédicat fabriqué par « eux-mêmes » à partir de $P(x, y)$ est $P(x, x)$*
certains	$(S / (sn \setminus S)) / n = C$	*« certains » (sujet) attend à droite un nom puis une phrase à laquelle il manque un sujet pour donner une phrase.*
	$(e \to t) \to ((e \to t) \to t) = C^*$	*étant donné un prédicat à une place P (nom commun) et un prédicat à une place Q (groupe verbal) « certains » fabrique une formule close*
	$\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge^{t\to(t\to t)}(P\ x)(Q\ x))))$	*la formule fabriquée par « certains » est $\exists x P(x) \wedge Q(x)$*

0. Analyse syntaxique

Il faut dériver le séquent suivant dans le calcul de Lambek en déduction naturelle :

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

En utilisant les abréviations C, E, P, X, S pour les catégories syntaxiques, voici la dérivation ou analyse syntaxique :

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

1. Calcul de l'ossature sémantique de l'énoncé

On applique le morphisme des catégories syntaxiques vers les types sémantiques, ce qui nous donne une déduction naturelle intuitionniste :

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

On extrait, par étiquetage, un λ-terme sur la conclusion du séquent final, λ-terme qui a une variable libre de chacun des types C^*, E^*, P^*, X^* :

$$((\mathsf{c}^{C^*}\,\mathsf{e}^{E^*})(\mathsf{x}^{X^*}\,\mathsf{p}^{P^*}))^{t}$$

3. Remplacement des variables par les λ-termes du lexique

On remplace chaque variable par le λ-terme sémantique correspondant, qui est fourni par le lexique et qui a le même type, ce qui donne le premier λ-terme de l'item suivant (il s'étend sur deux lignes).

4. Calcul de la sémantique par β-réduction

$$\begin{array}{c}\Big(\big(\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge(P\ x)(Q\ x)))))\big)\big(\lambda x^{e}(\mathtt{enonce}^{e\to t}\ x)\big)\Big)\\ \Big(\big(\lambda P^{e\to(e\to t)}\ \lambda x^{e}\ ((P\ x)x)\big)\big(\lambda y^{e}\ \lambda x^{e}\ ((\mathtt{parler\_de}^{e\to(e\to t)}\ x)y)\big)\Big)\end{array}$$

$$\downarrow \beta$$

$$\begin{array}{c}\big(\lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge^{t\to(t\to t)}(\mathtt{enonce}^{e\to t}\ x)(Q\ x)))))\\ \big(\lambda x^{e}\ ((\mathtt{parler\_de}^{e\to(e\to t)}\ x)x)\big)\end{array}$$

$$\downarrow \beta$$

$$\big(\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge(\mathtt{enonce}^{e\to t}\ x)((\mathtt{parler\_de}^{e\to(e\to t)}\ x)x)))\big)$$

Cette formule écrite dans le λ-calcul s'écrit habituellement :

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

Ce qui est la formule souhaitée.

Un exemple classique est de traiter la phrase suivante qui contient deux quantificateurs et peut être interprétée de deux manières.[7]

(6) Les enfants prendront une pizza.

(7) $\forall x \quad (\mathit{enfant}(x) \Rightarrow (\exists y \quad \mathit{pizza}(y) \wedge \mathit{prendre{-}futur}(x, y)\ )\ )$

(8) $\exists y \quad (\mathit{pizza}(y) \wedge \forall x \quad (\mathit{enfant}(x) \Rightarrow \mathit{prendre{-}futur}(x, y)\ )\ )$

Les grammaires catégorielles rendent compte de ce phénomène en produisant deux analyses syntaxiques, chacune produisant l'une des deux lectures. Mais c'est encore plus long que l'exemple développé ci-dessus.

4.4. Questions classiques de sémantique formelle

Les quantificateurs de notre langue sont plus riches et plus variés que ceux de la logique du premier ordre : *la plupart, les, un grand nombre de, peu de, etc.*

(9) La plupart des politiciens ont lu un livre d'économie.

Les *nombres* sont aussi des sortes de quantificateurs, en ce sens qu'ils lient des variables et posent aussi des problèmes de portée :

(10) Mettre huit gouttes dans trois cuillerées à soupe d'eau.

(11) 3x8=24 gouttes ?

(12) 8 gouttes ?

Se posent aussi des questions de portée entre les verbes de croyance et les quantificateurs qui peuvent figurer dans la complétive.

(13) James Bond croit que l'un des chercheurs du laboratoire est un espion.

(14) James Bond pense que Blofeld est un espion.

(15) James Bond a trouvé un microfilm dans le laboratoire.

Cette distinction, connue sous le nom de lecture *de re* (14) et lecture *de dicto*, (15) est due à Thomas d'Aquin, fondateur de l'université de Paris, même si le nom est du à Guillaume d'Ockham [35]. Du point de vue de la sémantique dite vériconditionnelle, on identifie le sens d'un énoncé à l'ensemble des conditions qui lui confèrent la valeur VRAI, c'est-à-dire la classe des mondes possibles dans lesquels il vaut VRAI [26].

(16) Cet étudiant croit que Chomsky est informaticien.

On interprétera la phrase ci-dessus ainsi : dans tous les mondes possibles compatibles avec les croyances de cet étudiant, Chomsky est un informaticien. La vérité de cette phrase dépend bien sûr de celle de la proposition subordonnée mais ne dépend pas du fait que Chosmky soit ou non informaticien, en accord notre intuition du sens de « croire » – voir par exemple [26].

Le principe Fregéen selon lequel le sens du tout est construit à partir du sens des parties connaît peu d'exceptions : en général on compose le sens des parties de la phrase en suivant sa structure :

(17) Dieu qui est invisible a créé le monde visible.

(18) Le chauffeur, qui n'a pas bu, raccompagnera les convives.

[7] On remarquera qu'il y a de subtiles différences entre les différentes formes de quantification ($\forall$: chaque, tout, tous, les, ... ; $\exists$: un, des, certains, ...) qui influent sur notre préférence entre les deux lectures. On notera aussi que nous avons une préférence pour que le premier quantificateur énoncé ait la plus grande portée, en comparant la phrase ci-dessus avec la suivante : *une pizza sera servie aux enfants*.

(19) Le chauffeur qui n'a pas bu raccompagnera les convives.

Néanmoins, ce principe de compositionnalité peut être mis en défaut, notamment à cause des pronoms qui montrent une divergence entre la syntaxe de la langue et celle de la logique. Prenons pour exemple la phrase suivante.

(20) Si une paysanne possède un âne, alors elle le bat.

(21) Si $(\exists p\ \exists a \quad \mathit{Ane}(a) \wedge \mathit{Paysanne}(p) \wedge \mathit{Possède}(p,a))$ alors $\mathit{Bat}(p,a)$.
Le $2^e a$ et le $2^e p$ sont libres !

Il s'agit d'une implication entre deux propositions, mais si le sens de la phrase est que le sens de la première implique le sens de la seconde, alors la seconde contient des variables libres alors qu'elles devraient être liées par les quantificateurs existentiels implicites dans les articles indéfinis de la première proposition, les deux *un*.

La distinction entre le langage formel de la logique et langage naturel doit rester présente. En effet, si la modélisation oublie les phrases dont sont issues les formules logiques alors on peine à expliquer la différence d'acceptabilité entre les deux phrases suivantes, qui sont logiquement identiques.

(22) J'avais trois trombones dans ma poche, je les ai tous perdus sauf un. Je le range dans un tiroir.

(23) * J'avais trois trombones dans ma poche, j'en ai perdu deux. Je le range dans un tiroir.

Ce genre de question est bien résolu par la *Discourse representation Theory* (DRT) [36, 23].

5. Modèles de l'acquisition de la syntaxe

Outre le calcul de représentations sémantiques, l'un des avantages des grammaires catégorielles que nous avons annoncés est l'existence d'algorithmes qui permettent à partir d'exemples de phrases de construire un lexique catégoriel (donc une grammaire, puisque les règles sont universelles) qui engendre ces phrases. Il s'agit donc de modéliser le processus naturel d'acquisition de la grammaire par le jeune enfant dont nous avons déjà parlé au paragraphe 3.3 du premier volet mais aussi d'acquérir automatiquement une grammaire à partir de données plus ou moins structurées.

Précisons les données du problème. Étant donnée une classe de grammaires, une fonction d'apprentissage φ est une fonction qui associe à tout ensemble fini d'exemples de phrases (de suite finies d'éléments de mots supposées être des phrases correctes) une grammaire de la classe (pour les grammaire catégorielles, un lexique, puisque les règles sont universelles, il n'y a pas à les apprendre). Il faut néanmoins un critère de convergence, nous utiliserons celui de E. Mark Gold aussi appelé apprentissage à la limite. Une classe de langages est dite apprenable à la limite s'il existe une fonction d'apprentissage telle que pour toute énumération $s_1, s_2, \ldots$ d'un langage $\mathbb{L}$ engendré par une grammaire de la classe, il existe un entier N tel que pour tout $n \geqslant N$, $\varphi(\{s_1, \ldots, s_N, \ldots, s_n\}) = \varphi(\{s_1, \ldots, s_N\})$ et $L(\varphi(\{s_1, \ldots, s_N\})) = \mathbb{L}$ – en notant $L(G)$ le langage engendré par une grammaire G. En français, cela signifie qu'au delà d'un certain nombre d'exemples, la fonction d'apprentissage propose toujours la même grammaire, et que celle-ci engendre exactement le langage qui est énuméré. [30]

Le cadre le plus simple pour présenter un algorithme d'apprentissage est le suivant : la classe de grammaires catégorielles considérée est celle des grammaires AB et les exemples de phrases sont les arbres de dérivation munis des noms des règles mais privés des catégories. Nous commenterons plus loin cette dernière restriction qui présuppose d'alimenter l'algorithme par des exemples bien plus structurés que de simples suites de mots.

Il est aisé de trouver des catégories syntaxiques pour chaque mot qui permettent de dériver les exemples. Nous savons que la racine d'un arbre de dérivation est S et en introduisant une nouvelle catégorie de base à chaque règle d'élimination, on peut facilement arriver à reconstruire une catégorie à chaque feuille de l'arbre, et donc à chaque mot. Si la conclusion de la règle est u, que la règle est $/_e$ (resp. $\backslash_e$) alors le sous arbre de droite (resp. de gauche) a pour racine x, une nouvelle catégorie de base, et celui de gauche $u \,/\, x$ (resp. celui de droite $x \setminus u$).

Cependant, si un mot a plusieurs occurrences dans les exemples proposés et que celles-ci requièrent des catégories syntaxiques différentes, on perd alors tout espoir de convergence au sens de Gold puisque, dans le lexique, on ajoutera sans fin des catégories syntaxiques. C'est pourquoi on limite le nombre de catégories par mot et le plus facile est de se limiter à *une* catégorie par mot : les grammaires ainsi faites sont dites *rigides* – là encore nous reviendrons sur cette restriction à la fin de cette section.

Afin de ne pas bloquer à chaque nouvelle occurrence d'un mot déjà vu, l'algorithme va chercher une catégorie syntaxique qui permette d'effectuer les dérivations autorisées par la catégorie syntaxique déjà présente dans le lexique ainsi que la dérivation associée au nouvel exemple. Une *substitution* σ est une fonction des catégories de base (hormis S) dans les catégories qui s'étend trivialement à l'ensemble des catégories ainsi :

$$\sigma(S) = S \qquad \sigma(A \setminus B) = \sigma(A) \setminus \sigma(B) \qquad \sigma(B \,/\, A) = \sigma(B) \,/\, \sigma(A)$$

Une substitution σ unifie les catégories $(A_i)_{i\in I}$ lorsque pour tout $i \in I$ $\sigma(A_i) = \sigma(A_1)$. Lorsqu'il existe une telle substitution appelée *unificateur*, il en existe une, disons σ_u, appelée unificateur principal, qui a la propriété suivante : si τ est un autre unificateur alors il existe une substitution σ_τ telle que $\tau = \sigma_\tau \circ \sigma_u$. On remarque que l'unification peut échouer, par exemple $a \setminus b$ et $x \,/\, y$ ne peuvent s'unifier car leur symbole principal n'est pas le même.

L'algorithme d'apprentissage *RG* (*Rigid Grammar*) proposé par Wojciech Buzkowsky et Gerald Penn [18] est relativement naturel :

(1) Associer comme indiqué un peu plus haut une catégorie à chaque mot de sorte que les exemples structurés $s_1, \ldots, s_n$ aient bien tous pour racine S.

(2) Unifier les catégories associées à un même mot. Si cela est possible, le lexique associé par *RG* à $s_1, \ldots, s_n$ est le lexique obtenu par l'unificateur principal (qui a bien une seule catégorie par mot).

La propriété de l'unification essentielle à l'apprentissage est une simple remarque : si un lexique *Lex* permet une dérivation d alors tout lexique obtenu par substitution à partir de *Lex* permet aussi la dérivation d. On peut donc, s'il existe, pour chaque mot, un unificateur de toutes les catégories correspondant à ses différentes occurrences, dériver tous les exemples avec cette unique catégorie obtenue par unification.

Si on considère les exemples structurés suivants :

(24) $[_{\backslash_e}[_{/_e}$ cette anguille $]$ nage $]$

(25) $[_{\backslash_e}[_{/_e}$ cette loutre $][_{\backslash_e}$ nage vite $]]$

Les catégories obtenues sont les suivantes en ajoutant des catégories de base comme expliqué plus haut :

(26) $[^{S}_{\backslash_e}[^{x_2}_{/_e}$ cette $:(x_2 \,/\, x_1)$ anguille $:x_1]$ nage $:(x_2 \setminus S)]$

(27) $[^{S}_{\backslash_e}[^{y_2}_{/_e}$ cette $:(y_2 \,/\, y_3)$ loutre $:y_3][^{(y_2 \setminus S)}_{\backslash_e}$ nage $:y_1$ vite $:(y_1 \setminus (y_2 \setminus S))]]$

Les catégories collectées sont alors :

mot	catégorie 1	catégorie 2
cette :	$x_2 \,/\, x_1$	$y_2 \,/\, y_3$
vite :		$y_1 \setminus (y_2 \setminus S)$
anguille :	x_1	
loutre :		y_3
nage :	$x_2 \setminus S$	y_1

Il faut maintenant unifier toutes les catégories associées à chaque mot, et dans ce cas c'est possible :

(28)

$$\begin{aligned} \sigma_u(x_1) &= z_1 \\ \sigma_u(x_2) &= z_2 \\ \sigma_u(y_1) &= z_2 \setminus S \\ \sigma_u(y_2) &= z_2 \\ \sigma_u(y_3) &= z_1 \end{aligned}$$

et cela conduit à la grammaire catégorielle rigide suivante :

(29)

cette :	$z_2 \,/\, z_1$
vite :	$(z_2 \setminus S) \setminus (z_2 \setminus S)$
anguille :	z_1
loutre :	z_1
nage :	$z_2 \setminus S$

L'algorithme est assez aisé à comprendre mais la preuve de sa convergence due à Makoto Kanazawa n'a rien d'évident [34]. Schématiquement, l'unification construit une suite croissante de grammaires pour un certain ordre et cette suite est majorée par la grammaire ayant engendré le langage énuméré.

Théorème 5.1 (Buszkowski, Penn, Kanazawa)**.** *Les grammaires catégorielles AB avec au plus une catégorie par mot sont apprenables au sens de Gold à partir d'arbres d'analyse incomplets.*

Ce résultat appelle quelques commentaires. Observons déjà qu'il ne contredit pas le résultat de Gold publié dans le même article [30] qui affirme que toute classe contenant les langages réguliers n'est pas apprenable : les grammaires AB rigides constituent une classe transverse à la hiérarchie de Chomsky, qui engendre des langages algébriques (produits par une grammaire non contextuelle) non réguliers, mais pas tous les langages réguliers.

On remarque aussi qu'on apprend à partir de structures (des arbres étiquetés). On peut se passer des étiquettes, voire des arbres, mais alors il faut les essayer

toutes et modifier le critère de convergence. La fonction d'apprentissage associe un ensemble de grammaires aux exemples vus, et le critère de convergence devient : au-delà d'un certain nombre d'exemples cet ensemble contient toujours la grammaire solution.

Une autre contrainte est de n'avoir qu'une seule catégorie par mot (les grammaires apprises sont rigides). D'un point de vue pratique, dans une langue à ordre des mots assez fixe, les modèles de Markov cachés du premier volet section 2.4 réussissent fort bien à distinguer les occurrences d'un même mot nécessitant des catégories syntaxiques distinctes, comme *le* article et *le* pronom, ou comme *manger* dans un usage absolu ou transitif (*Que fait-il ? Il mange.* par opposition à *Il mange une pomme.*), auquel cas on peut supposer que les grammaires à apprendre sont effectivement rigides, puisqu'on distingue *mange : vt* $= (sn \setminus S) / sn$ de *mange : vi* $= (sn \setminus S)$.

En revanche, d'un point de vue théorique, comment faire ? On peut, comme l'a fait Kanazawa, chercher à unifier en gardant toujours moins de k catégories par mot. L'algorithme consiste à essayer les possibilités et procède donc, comme précédemment avec des ensembles de solutions.

Une solution plus astucieuse consiste à contraindre la forme des catégories. Effectivement, est apprenable la classe des grammaires dont les catégories ne contiennent pas de connecteurs en position négative (ce qui ne limite pas la capacité générative) et diffèrent toujours en plus d'un endroit pour un même mot (ce qui limite la capacité générative et correspond à une notion de réversibilité).[11]

Finalement, concernant les grammaires de Lambek, nous avons montré avec Roberto Bonato qu'elles sont apprenables à partir des structures de dérivation décrites en section 3.2. C'est surtout la partie assignation de catégories aux mots qui diffère, elle ressemble à l'algorithme de calcul du type simple le plus général d'un λ-terme. [13]

Sur ce sujet on pourra consulter une synthèse récente et peu technique [7].

6. Grammaires catégorielles et théorie des langages standard

6.1. Complétude de l'interprétation du calcul de Lambek dans les monoïdes libres

Une bonne logique, outre un système déductif élégant et symétrique, se doit d'avoir une notion de modèle aussi naturelle que possible et d'être complète pour cette notion de modèle. Par exemple la notion naturelle de modèle pour le calcul propositionnel classique est l'algèbre de Boole dont un représentant particulièrement simple est $\{\text{FAUX}, \text{VRAI}\}$, modèle pour lequel il y a complétude : une proposition qui vaut VRAI quelles que soient les valeurs assignées aux variables propositionnelles (l'analogue des catégories de base) est démontrable. Nous donnerons ici l'analogue de l'algèbre de Boole pour le calcul de Lambek, les semi-groupes résidués, et en donnerons deux exemples naturels : le groupe libre (qui n'est pas complet) et les parties d'un semi-groupe libre, pour lequel il y a complétude en partant d'un cas particulier appelé semi-groupe syntaxique. On pourrait ainsi dire que le calcul de Lambek est *le* système déductif adapté à décrire le semi-groupe libre.

Un modèle est un ensemble ordonné muni de trois opérations qui correspondent aux implications $\backslash$ et $/$ et à la virgule qui concatène les hypothèses.

Un *semi-groupe résidué*, est une structure $(M, \circ, \backslash\backslash, //, \sqsubset)$où

– M est un ensemble
– $\circ$ est une loi de composition associative sur M ($(M, \circ)$ est un semi-groupe)
– $\backslash\backslash$ et $//$ sont deux lois de composition sur M.
– $\sqsubset$ est un ordre (partiel) M.

qui satisfait la propriété suivante :

(RSG) Quels que soient les éléments a, b, c de M les relations suivantes sont équivalentes :

$$\begin{array}{rcl} a & \sqsubset & (c // b) \\ (a \circ b) & \sqsubset & c \\ b & \sqsubset & (a \backslash\backslash c) \end{array}$$

Une interprétation est la donnée, pour chaque catégorie de base de son interprétation comme élément de M, laquelle sera notée $[p]$. L'interprétation s'étend alors trivialement à toute catégorie :

$$[A \setminus B] = [A] \backslash\backslash [B] \qquad [B / A] = [B] // [A]$$

ainsi qu'aux suites de catégories :

$$[A_1, \ldots A_n] = [A_1] \circ \cdots \circ [A_n]$$

Un séquent $A_1, \ldots, A_p \vdash C$ est dit valide dans le modèle pour l'interprétation $[\,]$ si et seulement si $[A_1, \ldots, A_p] \sqsubset [C]$ – tout comme, dans une algèbre de Boole, $A \vdash B$ est VRAI lorsque que $[A] \leqslant [B]$. Donnons de suite une propriété facile à démontrer par induction sur la hauteur de l'arbre de dérivation, qui montre le bien fondé de cette interprétation du calcul de Lambek dans les semi-groupes résidués :

Théorème 6.1 (Validité de l'interprétation dans les semi-groupes résidués)**.** *Si un séquent $A_1, \ldots, A_n \vdash C$ est démontrable alors quels que soient le semi-groupe résidué et l'interprétation choisie, le séquent est valide, c'est-à-dire $[A_1, \ldots, A_p] \sqsubset [C]$.*

Donnons quelques exemples de monoïdes résidués pour lesquels la vérification de (RSG) est aisée.

Le modèle du groupe libre est défini par :

– $(M, \cdot)$ est le groupe libre sur les catégories de base
– $a \backslash\backslash b$ est $(a)^{-1}b$
– $b // a$ est $b(a)^{-1}$
– $a \sqsubset b$ est $a = b$ (ordre discret)

On peut bien évidemment utiliser l'interprétation $[p] = p$, et c'est l'interprétation standard. On calcule l'interprétation d'une formule A au moyen de $[p] = p$, de $[A \setminus B] = [A]^{-1}[B]$ et de $[B / A] = [B][A]^{-1}$ et des équations $(AB)^{-1} = B^{-1}A^{-1}$, $(A^{-1})^{-1} = A$ et $xx^{-1} = x^{-1}x = 1$. L'interprétation d'une suite de formules est le produit des interprétations des formules qui la composent. Un séquent $A_1, \ldots, A_n \vdash C$ est valide dans ce modèle si et seulement si $[A_1] \ldots [A_n] = [C]$ c'est-à-dire $[A_n]^{-1} \cdots [A_1]^{-1}[C] = 1$.

On notera que ce modèle n'est pas complet : une raison intuitive est qu'il traduit $A \vdash B$ qui n'est pas une relation symétrique par $[A] = [B]$ qui est symétrique. Or

dans le calcul de Lambek il y a bien des couples de catégories A, B telles que $A \vdash B$ sans que $B \vdash A$, par exemple $x \vdash (a \,/\, x) \setminus a$ mais $(a \,/\, x) \setminus a \not\vdash x$.

Le modèle des parties d'un semi-groupe est défini par la donnée un semi-groupe $(W, \cdot)$, éventuellement libre :

- $M = 2^W$
- $A \circ B = \{a \cdot b \quad | \quad a \in A \text{ and } b \in B\}$
- $A \setminus\!\setminus B = \{z \quad | \quad \forall a \in A \quad a \cdot z \in B\}$ (*)
- $B \,/\!/\, A = \{z \quad | \quad \forall a \in A \quad z \cdot a \in B\}$ (**)
- $A \sqsubset B$ lorsque $A \subset B$ (inclusion d'ensembles).

Le semi-groupe syntaxique est définie comme précédemment mais le semi-groupe dont on considère les parties est le semi-groupe libre des suites non vides de formules. L'interprétation standard est $[p] = \{\Gamma \mid \Gamma \vdash p \text{ est démontrable}\}$. On peut vérifier que l'extension de l'interprétation aux formules par (*) et (**) ci-dessus préserve cette propriété : par récurrence sur F on montre que pour toute catégorie F on a $[F] = \{\Gamma \mid \Gamma \vdash F \text{est démontrable}\}$ (1), et par conséquent on a toujours $F \in [F]$ (2).

Théorème 6.2 (Buszkowski)**.** *Un séquent valide dans tout semi-groupe résidué pour toute interprétation est démontrable dans le calcul de Lambek.* [16, 17]

Démonstration. Un tel séquent $A_1, \ldots, A_p \vdash C$ est en particulier valide dans le semi-groupe syntaxique pour l'interprétation standard. Par (2) on a $A_i \in [A_i]$ et donc $A_1, \ldots, A_n \in [A_1, \ldots, A_n]$ et comme le séquent est valide $[A_1, \ldots, A_p] \sqsubset [C]$, c'est-à-dire $[A_1, \ldots, A_p] \subset [C]$ et donc $A_1, \ldots, A_n \in [F]$. D'après (1) cela signifie que $A_1, \ldots, A_p \vdash C$ est démontrable. □

6.2. Les grammaires AB sont non contextuelles

Théorème 6.3 (Bar-Hillel, Gaifman, Shamir)**.** *Les grammaires AB sont faiblement équivalentes aux grammaires non contextuelles : elles engendrent la même classe de langages.* [10]

Démonstration. Étant donnée une grammaire catégorielle Lex, construisons une grammaire non contextuelle G, en forme normale de Chomsky (voir premier volet théorème 3.2) qui engendre le même langage. Les terminaux sont les mêmes, c'est-à-dire les mots du lexique. Les non terminaux N sont toutes les catégories du lexique et toutes leurs sous catégories au sens du paragraphe 2.1 et le symbole de départ est S, le même. Quant aux règles de production, elles contiennent les règles $V \to U(U \setminus V)$ et $V \to (V \,/\, U)U$ pour tous les non terminaux U, V de N (en nombre fini) ainsi que les règles $C \to a$ lorsque $C \in \mathrm{Lex}(a)$ (en nombre fini également).

Réciproquement, étant donnée une grammaire non contextuelle que l'on peut sans perte de généralité du point de vue du langage engendré supposer en forme normale de Greibach (voir le théorème 3.3 du premier volet), c'est-à-dire avec des règles de la forme $X \to aX_1 \ldots X_p$, on peut définir une grammaire catégorielle qui engendre le même langage : les mots sont les terminaux, les catégorie de bases sont les non terminaux et pour chaque règle $X \to aX_1 \ldots X_p$ on ajoute la catégorie $(((X \,/\, X_p) \,/\, X_{p-1}) \cdots \,/\, X_1$ à l'ensemble de catégories $\mathrm{Lex}(a)$. □

6.3. La conjecture de Chomsky (63) et sa résolution par Pentus (93) : les grammaires de Lambek sont non contextuelles

Avant de voir que les grammaires de Lambek sont non contextuelles, assurons nous déjà que pour toute grammaire non contextuelle il existe une grammaire de Lambek engendrant le même langage. Nous venons de voir qu'une grammaire non contextuelle est équivalente à une grammaire AB qui n'utilise que des catégories de la forme $(((X \,/\, X_p) \,/\, X_{p-1}) \cdots) \,/\, X_1$. Mais avec la propriété de la sous formule dont nous avons parlé à la section 3.2, il n'est pas très difficile de voir qu'avec des catégories de cette forme, AB et le calcul de Lambek dérivent les mêmes séquents. Le même lexique engendre donc le même langage, celui de la grammaire non contextuelle initiale. Ce résultat a été établi par Joel M. Cohen dans [22] (mathématicien qui s'est ensuite consacré à la topologie algébrique).

En fait cet article contenait aussi une preuve fausse de la réciproque, problème proposé par Noam Chomsky dans [20] où il conjecture que les grammaires de Lambek n'engendrent que des langages algébriques, c'est-à-dire que les grammaires de Lambek sont faiblement équivalentes aux grammaires non contextuelles. *A posteriori* il est difficile de dire s'il s'agit d'une intuition géniale ou d'une remarque en l'air. Toujours est-il que c'est vrai. Wojciech Buszkowski en 1982 [15] a découvert l'erreur de [22] et Mati Pentus, en 1992 a effectivement établi cette conjecture [52]. C'est un joli résultat qui utilise des notions récentes de la logique linéaire [27] et un raffinement du théorème d'interpolation de Dirk Roorda [62, 63] qui n'existait pas à l'époque de Lambek. On utilisera ici le calcul des séquents plutôt que la déduction naturelle, car la règle de coupure jouera un rôle essentiel – même si elle n'augmente pas la classe des tautologies.

Théorème 6.4 (Pentus)**.** *Les grammaires de Lambek sont faiblement équivalentes aux grammaires non contextuelles : pour toute grammaire de Lambek, il existe une grammaire non contextuelle qui engendre le même langage.*

Plutôt que la preuve complète nous donnons ici les lemmes, une idée de leurs preuves (sauf pour celui sur le groupe libre, qui, en fait, était déjà connu), et la preuve du résultat principal à partir des lemmes. On note $|A|$ la taille d'une catégorie c'est-à-dire son nombre d'occurrences de catégories de bases (par exemple $|(a \setminus b) \,/\, a| = 3$).

Lemme 6.5. *Si un séquent $A_1, \ldots, A_n \vdash A_{n+1}$ avec $|A_i| \leqslant K$ alors il est démontrable avec uniquement la règle de coupure à partir de séquents démontrables $U, V \vdash X$ ou $U \vdash X$ avec $|U|, |V|, |X| \leqslant K$.*

On remarquera que l'ensemble S_K des séquents démontrables $U, V \vdash X$ et $U \vdash X$ avec $|U|, |V|, |X| \leqslant K$ est fini.

Démonstration de 6.5 entraîne 6.4. Considérons la grammaire non contextuelle suivante dont les mots (les terminaux) sont les mêmes, dont les non terminaux sont les catégories de taille inférieure à K et dont les règles sont $X \to UV$ si $(U, V \vdash X) \in \mathsf{S}_K$ et $X \to U$ si $(U \vdash X) \in \mathsf{S}_K$.

Si une phrase $m_1 \ldots m_n$ est dans le langage de la grammaire catégorielle, alors elle appartient au langage de la grammaire non contextuelle. Prenons K la taille de la plus grande catégorie du lexique. Comme $m_1 \ldots m_n$ est dans le langage de la grammaire catégorielle, il existe des catégories $t_i \in \mathrm{Lex}(m_i)$ (et donc de taille

inférieure à K) une déduction de $t_1, \ldots, t_n \vdash S$. D'après le lemme 6.5, on peut supposer que cette déduction n'est composée que de séquents de S_K et de coupures, c'est-à-dire de règles de la grammaire non contextuelle et de la composition de règles. On obtient donc une dérivation de $S \to t_1 \cdots t_n$ dans la grammaire non contextuelle, dérivation qui, poursuivie par les règles lexicales, donne une dérivation de $S \to m_1 \cdots m_n$.

Si une phrase appartient au langage de la grammaire non contextuelle, alors elle appartient au langage de la grammaire catégorielle. Une dérivation avec cette grammaire non contextuelle peut-être divisée en deux parties : d'une part l'application des règles (en commençant par une règle $S \to \ldots$) qui correspondent aux séquents de S_K et la composition de ces règles (qui correspond à la coupure) et d'autre part les règles lexicales qui réécrivent un non terminal en un terminal. La première partie, donne, *mutatis mutandis*, une preuve formelle de $t_1, \ldots, t_n \vdash S$ n'utilisant que des coupures, et comme $t_i \in \mathrm{Lex}(m_i)$, la phrase $m_1 \cdots m_n$ appartient au langage de la grammaire catégorielle. □

Le lemme 6.5 qu'il reste à prouver a une allure bien connue des logiciens : il s'agit d'un lemme d'interpolation, et pour des séquents dits *minces* on sait majorer précisément la taille de l'interpolant. Les séquents minces sont des séquents démontrables où toute catégorie de base apparaît exactement zéro ou deux fois. Il est facile de voir que toute démonstration dans le calcul de Lambek s'obtient par substitution lettre à lettre de catégories de base dans une démonstration ne comportant que des séquents minces, comme on peut l'observer dans les exemples de dérivations du paragraphe 3.3. Il en résulte que tout séquent démontrable est le résultat d'une substitution lettre à lettre dans d'un séquent mince.

Lemme 6.6 (Interpolation pour les séquents minces)**.** *Soit $\Gamma, \Delta, \Theta \vdash C$ un séquent mince alors il existe une formule B (un interpolant de Δ), tel que :*

(1) $\Delta \vdash B$ *est mince*

(2) $\Gamma, B, \Theta \vdash C$ *est mince*

(3) $|B| = \|[\Delta]\|$ *– le nombre d'occurrences de catégories de base dans B est égal à la taille de l'interprétation de Δ dans le groupe libre.*

On remarquera que (1) et (2) entraînent $\Gamma, \Delta, \Theta \vdash C$ à l'aide de la règle de coupure. On peut combiner ce résultat avec une propriété sur la taille notée $||...||$ des mots réduits du groupe libre redécouverte par Pentus mais qui se trouvait déjà dans [51, 2].

Lemme 6.7 (Une propriété du groupe libre)**.** *Soit u_i $i = 1, \ldots, n+1$ des éléments du groupe libre tels que $u_1 \cdot \cdots \cdot u_{n+1} = 1$ alors il existe $I \leqslant n$ tel que*

$$\|u_I u_{I+1}\| \leqslant \max(\|u_I\|, \|u_{I+1}\|)$$

Démonstration de 6.6 et 6.7 entraînent 6.5. Comme la taille des formules et des séquents n'est pas modifiée par les substitutions lettre à lettre, il suffit de montrer cela pour les séquents minces. En effet, si tout séquent mince se déduit de tels séquents, il en est de même des séquents non minces, il suffit de faire une substitution qui ne change ni la forme des séquents, ni la taille des formules.

Pour montrer ce résultat pour les séquents minces, on procède par récurrence sur le nombre de formules dans le séquent. Si $n \leqslant 2$, il n'y a rien à démontrer.

Sinon, comme il s'agit d'un séquent démontrable, l'interprétation canonique dans le groupe libre du paragraphe 6.1 nous dit que $[A_1]....[A_n][A_{n+1}]^{-1} = 1$.

En utilisant la propriété 6.7 et le fait que la taille de chaque formule et donc de chaque $[A_i]$ soit moins que K, il existe deux formules consécutives $A_l A_{l+1}$ (le cas où se sont les deux dernières nécessite une petite adaptation) telles que la taille dans le groupe libre du mot réduit $[A_l][A_{l+1}]$ soit moins que K. En utilisant le lemme d'interpolation 6.6 on trouve donc un interpolant de taille $\leqslant K$, et donc deux séquents minces (démontrables) l'un ayant deux formules à gauche et l'autre $n-1$ et dont toutes les formules sont de taille $\leqslant K$. Par hypothèse de récurrence chacun des deux est démontrable à partir de séquents de la forme voulue, et donc aussi le séquent mince initial qui s'obtient à partir d'eux par une simple règle de coupure. □

6.4. Analyses catégorielles et langages de graphes et d'arbres

Dans la mesure où, comme nous le mentionnions dans le premier volet, la théorie des langages s'intéresse depuis les années soixante-dix aux grammaires d'arbres et depuis la fin des années quatre-vingt aux langages de graphes, il est normal de se demander où se situent les analyses catégorielles, qui sont des arbres ou des graphes, dans ce type de hiérarchies. Le premier résultat, dû à Hans-Jorg Tiede [66] a montré que les arbres de dérivation d'une grammaire de Lambek (les arbres correspondant à des déductions naturelles normales) constituent un langage d'arbres qui peut ne pas être régulier – les langages réguliers d'arbres se reconnaissent par des machines similaires aux transducteurs définis dans le premier volet. C'est étonnant, car les chaînes reconnues par ces deux types de grammaires sont les mêmes, les langages non contextuels, et les arbres d'analyses des grammaires non contextuels sont réguliers (et même locaux). Il a même été montré tout récemment par Makoto Kanazawa et Sylvain Salvati que les langages des arbres de dérivation d'une grammaire de Lambek, ou les réseaux de démonstration, ne sont pas des langages d'arbres non contextuels, mais sont dans une classe incomparable : ils appartiennent à la classe des langages d'arbres obtenus par des grammaires de graphes dites à remplacement d'hyperarêtes (HR grammars). Étonnant pour des grammaires dont les langages de chaînes sont non contextuels, lesquels correspondent usuellement à des langages réguliers d'arbres ! Les résultats en cours sur les réseaux de démonstration du paragraphe 3.3 semblent similaires.

Nous n'avons pas formellement défini la variante non associative des grammaires de Lambek, due au même auteur [40], mais on peut en donner une idée rapidement. Au lieu d'être une simple suite de formules, la partie gauche d'un séquent est un arbre binaire de formule, qu'on peut noter avec des $\langle , \rangle$. Les règles d'élimination fabriquent de tels arbres (par exemple la conclusion de la règle $\backslash_e$ serait : $\langle \Gamma, \Delta \rangle$) et pour pouvoir appliquer une règle d'introduction il faut que l'hypothèse du séquent soit une fille de la racine (par exemple $\langle A, \Gamma \rangle$ pour la règle $\backslash_i$). Ces grammaires engendrent aussi tous les langages produits par des grammaires non contextuelles (elles contiennent clairement les grammaires AB) et le calcul a une complexité polynomiale (et donc ces grammaires ont une analyse polynomiale). Les grammaires de Lambek non associatives ont effectivement des langages d'arbres d'analyse réguliers

c'est-à-dire qu'ils sont, à renommage près, pas plus compliqués que ceux des grammaires non contextuelles : c'est moins surprenant. [61]

7. Grammaires catégorielles d'hier et d'aujourd'hui

À travers les grammaires catégorielles et leur présentation logique à l'aide du calcul de Lambek, on peut donc en même temps qu'on analyse une phrase construire sa structure sémantique dans la logique d'ordre supérieur. On peut aussi tirer parti de la lexicalisation pour les acquérir automatiquement à partir d'exemples positifs structurés ou de corpus annotés.

Mais au vu des résultats d'équivalence avec les grammaires hors-contextes, ces grammaires semblent trop limitées d'un point de vue syntaxique, puisque nous avons vu au paragraphe 3.2 du premier volet que les langues naturelles ont des syntaxes légèrement contextuelles. Effectivement, les grammaires de Lambek peinent à rendre compte de nombreuses constructions syntaxiques courantes : constituants discontinus (*ne ... pas*), pronoms clitiques dans les langues romanes (*Je la fais réparer. Je peux la réparer.*),... Pour ce faire, les travaux récents proposent deux types d'extensions qui préservent la correspondance entre la structure syntaxique et la structure logique de la phrase.

La première approche due à Michael Moortgat [47] consiste à enrichir la logique de base le calcul de Lambeck non associatif par des modalités (connecteurs unaires) et à utiliser des postulats qui régissent le comportement de ces modalités. On peut ainsi fabriquer des formules dont les constituants sont inaccessibles jusqu'à ce qu'ils interagissent avec une formule précédée de la modalité duale. On décrit ainsi tous les langages récursivement énumérables, et lorsqu'on se limite à certains type de postulats, on garde une analyse polynomiale. Cette approche a été développée jusqu'à des développements pratiques très conséquents, comme l'analyseur Grail de Richard Moot qui a utilisé un processus d'extraction automatique de la grammaire (du lexique) à partir d'un corpus correctement annoté. [48]

Une autre famille d'approches consiste à construire une structure profonde dans la logique linéaire commutative (une variante du calcul de Lambek qui identifie $\backslash$ et $/$) et à en déduire d'une part la forme de surface (l'ordre des mots) et d'autre part la forme logique. On peut rattacher à cette tendance les grammaires minimalises catégorielles d'Alain Lecomte et moi-même [41, 42] qui s'inspirent de la formalisation par Edward Stabler [64] du programme minimaliste de Noam Chomsky [21]. L'idée est de pouvoir reprendre les analyses minimalistes de nombreuses constructions syntaxiques et la variation qu'elles proposent entre phrases et même entre langues, et aussi d'aborder les questions de syntaxe et sémantique comme les coréférences possibles ou impossibles des pronoms [4, 12]. D'autres systèmes basés sur la même architecture sont apparus : les grammaires d'interaction de Guy Perrier [54], les grammaires catégorielles abstraites de Philippe de Groote [24], les *lambda grammars* de Reinhard Muskens [50], les *higher-order categorical grammars* de Carl Pollard [55].

Le lien entre théorie des langages standard et théorie de la démonstration conserve des mystères en dépit des résultats importants dont nous venons de parler. D'un point de vue algorithmique et donc pratique la théorie des langages est plus efficace, comme le montre le résultat de complexité, le théorème 3.2. Cependant, la richesse des preuves permet de construire des représentations

sémantiques et les grammaires catégorielles sont plus aisées à acquérir. Ce lien entre les deux approches, s'il était mieux connu, permettrait peut-être d'importer dans les grammaires catégorielles des algorithmes rapides ; et, utilisé en sens contraire, il pourrait permettre de compiler les grammaires catégorielles en grammaires de réécriture plus efficaces, en maintenant une correspondance qui permettrait de calculer des représentations du sens.

8. Conclusion

L'objet de cet article en deux volets était de présenter les méthodes mathématiques en linguistique computationnelle, et, après une présentation de la linguistique, nous avons exposé deux traditions relativement mathématiques pour décrire la structure du langage naturel, ou, plus particulièrement, celle des mots et de la phrase : celle issue de la théorie des langages et celle issue de la logique, avec des connexions prometteuses qui commencent à se tisser entre les deux.

Si on revient au panorama des domaines de la linguistique esquissé dans le premier volet, il est étonnant que la sémantique ne soit que très schématiquement évoquée par notre présentation et que l'énonciation ou la pragmatique ne le soient pas du tout.

En fait, du côté sémantique, il existe déjà bien plus que ce que nous avons présenté ici. Par exemple les questions mentionnées au paragraphe 4.4 sont à peu près toutes correctement modélisées, notamment par l'intensionnalité et les mondes possibles à la Kripke, ou par la *Discourse Representation Theory* de Hans Kamp [36]. On voit aussi apparaître des modèles relativement formels de la sémantique lexicale, qui relie les sens des mots et explore leur lien au monde. [57]

Cependant, même s'il existe des modèles de la sémantique plus fins que ceux que nous avons esquissés ici, d'un point de vue mathématique, ils sont encore instables et il n'y a guère de résultats mathématiques hormis sur des questions particulières comme les quantificateurs généralisés [38]. Cet état de fait est également frustrant d'un point de vue pratique : tandis qu'on sait analyser automatiquement de très gros volumes de textes en peu de temps, on ne sait pas interpréter les structures syntaxiques obtenues, alors que la recherche d'informations, le dialogue homme machine ou la traduction assistée par ordinateur bénéficieraient grandement de modèles calculatoires du sens. Du point de vue des sciences cognitives, il serait également intéressant de savoir comment nous nous comprenons à travers le langage, quel type de modèle mathématique correspond le mieux à cette activité, et peut-être quels fragments de quelles logiques nous utilisons – si toutefois la logique est la mieux adaptée à décrire ces questions de sens.

En se limitant aux considérations syntaxiques et à une sémantique rudimentaire dont les modèles mathématiques respectifs sont suffisamment développés, on espère avoir suscité l'intérêt du mathématicien pour un domaine relativement traditionnel, entre linguistique, mathématiques et informatique où il est à la fois possible de résoudre des problèmes ouverts mais aussi de construire des objets mathématiques, en particulier de nature logique ou de combinatoire, qui soient à la fois satisfaisants en tant qu'objets mathématiques mais aussi pertinents d'un point de vue linguistique.

9. Références

[1] V. M. Abrusci – « Phase semantics and sequent calculus for pure non-commutative classical linear logic », *Journal of Symbolic Logic* **56** (1991), n° 4, p. 1403–1451.

[2] J.-M. Autebert, L. Boasson & G. Sénizergues – « Langages de parenthèses, langages N.T.S. et homomorphismes inverses », *R.A.I.R.O. Informatique Théorique* **18** (1984), n° 4, p. 327–344.

[3] K. Ajdukiewicz – « Die syntaktische konnexität », *Studia Philosophica* **1** (1935), p. 1–27, (English translation in [44], pages 207–231).

[4] M. Amblard – « Calcul de représentations sémantiques et syntaxe générative : les Grammaires Minimalistes Catégorielles », Thèse, Université de Bordeaux I, 2007.

[5] A. Arnauld & C. Lancelot – *Grammaire générale et raisonnée*, Le Petit, 1660, Appelée *Grammaire de Port-Royal*. Rééditée en 1997 par les Éditions Allia.

[6] A. Arnauld & P. Nicole – *La logique ou l'art de penser : contenant outre les règles communes, plusieurs observations nouvelles, propres à former le jugement.*, G. Desprez, 1683, Réédité par Vrin en 2002.

[7] D. Bechet, R. Bonato, A. Dikovsky, A. Foret, Y. Le Nir, E. Moreau, C. Retoré & I. Tellier – « Modèles algorithmiques de l'acquisition de la syntaxe : concepts et méthodes, résultats et problèmes », *Recherches Linguistiques de Vincennes* **36** (2007), p. 123–152.

[8] M. Baratin & F. Desbordes – *L'analyse linguistique dans l'antiquité classique – I les théories*, Klincksieck, 1981.

[9] Y. Bar-Hillel – « A quasi arithmetical notation for syntactic description », *Language* **29** (1953), p. 47–58.

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

[11] J. Besombes & J.-Y. Marion – « Learning discrete categorial grammars from structures », *RAIRO, Theoretical Informatics and Applications* **42** (2008), p. 165–182.

[12] R. Bonato – « An integrated computational approach to binding theory », Tesi di dottorato di ricerca and thèse de doctorat, Università degli Studi di Verona and Université Bordeaux I, May 2006.

[13] R. Bonato & C. Retoré – « Learning rigid Lambek grammars and minimalist grammars from structured sentences », in *Proceedings of the third workshop on Learning Language in Logic, LLL 01* (Strabourg) (L. Popelìnskỳ & M. Nepil, éds), FI MU Report series, n° FI-MU-RS-2001-08, Faculty of Informatics – Masaryk University, September 2001, p. 23–34.

[14] J. Busquets – *Logique et langage : apports de la philosophie médiévale*, Presses Universitaires de Bordeaux, 2006.

[15] W. Buszkowski – « Compatibility of a categorial grammar with an asssociated category system », *Zeitschrift für matematische Logik und Grundlagen der Mathematik* **28** (1982), p. 229–238.

[16] ———, « Completeness results for Lambek syntactic calculus », *Zeitschrift für matematische Logik und Grundlagen der Mathematik* **32** (1986), p. 13–28.

[17] ———, « Mathematical linguistics and proof theory », in *Handbook of Logic and Language* [67], p. 683–736.

[18] W. Buszkowski & G. Penn – « Categorial grammars determined from linguistic data by unification », *Studia Logica* **49** (1990), p. 431–454.

[19] C. Casadio – « Semantic categories and the development of categorial grammars », in *Categorial Grammars and Natural Language Structures* (R. Oehrle, E. Bach & D. Wheeler, éds), Reidel, Dordrecht, 1988, p. 95–124.

[20] N. Chomsky – « Formal properties of grammars », in *Handbook of Mathematical Psychology*, vol. 2, Wiley, New-York, 1963, p. 323 – 418.

[21] ———, *The minimalist program*, MIT Press, Cambridge, MA, 1995.

[22] J. M. Cohen – « The equivalence of two concepts of categorial grammars », *Information and Control* **10** (1967), p. 475–484.

[23] F. Corblin – *Représentation du discours et sémantique formelle. introduction et applications au francais.*, Linguistique Nouvelle, Presses Universitaires de France (PUF), 2002.

[24] P. de Groote – « Abstract categorial grammars », in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL 2001* (Toulouse), ACL, July 2001.
[25] A. Finkel & I. Tellier – « A polynomial algorithm for the membership problem with categorial grammars », *Theoretical computer science* **164** (1996), n° 1–2, p. 207–221.
[26] L. T. F. Gamut – *Logic, language and meaning – volume 2 : Intensional logic and logical grammar*, The University of Chicago Press, 1991.
[27] J.-Y. Girard – « Linear logic », *Theoretical Computer Science* **50** (1987), n° 1, p. 1–102.
[28] J.-Y. Girard, Y. Lafont & P. Taylor – *Proofs and types*, Cambridge Tracts in Theoretical Computer Science, n° 7, Cambridge University Press, 1988.
[29] G. Gobber – « Alle origini della grammatica categoriale », *Rivista di Filosofia Neo-Scolastica* **77** (1985), p. 258–295.
[30] E. M. Gold – « Language identification in the limit », *Information and control* **10** (1967), p. 447–474.
[31] Howard, William A. – « The formulae-as-types notion of construction », To H.B. Curry : Essays on Combinatory Logic, λ-calculus and Formalism, J. Hindley and J. Seldin eds., Academic Press, p.479-490.
[32] E. Husserl – *Logische untersuchungen*, second éd., Max Niemeyer, 1913, Traduction française de Hubert Élie, Lothar Kelkel et René Schérer : Recherches Logiques, Presses Universitaires de France, 1962.
[33] T. M. V. Janssen – « Compositionnality », in *Handbook of Logic and Language* [67], p. 417–473.
[34] M. Kanazawa – *Learnable classes of categorial grammars*, Studies in Logic, Language and Information, FoLLI & CSLI, 1998, distributed by Cambridge University Press.
[35] W. Kneale & M. Kneale – *The development of logic*, 3rd éd., Oxford University Press, 1986.
[36] H. Kamp & U. Reyle – *From discourse to logic*, D. Reidel, Dordrecht, 1993.
[37] J.-L. Krivine – *Lambda calcul – types et modèles*, Etudes et Recherches en Informatique, Masson, Paris, 1990.
[38] E. L. Keenan & D. Westerstål – « Generalized quantifiers in linguistics and logic », in *Handbook of Logic and Language* [67], p. 837–894.
[39] J. Lambek – « The mathematics of sentence structure », *American mathematical monthly* (1958), p. 154–170.
[40] ———, « On the calculus of syntactic types », in *Structure of language and its mathematical aspects* (R. Jakobson, éd.), American Mathematical Society, 1961, p. 166–178.
[41] A. Lecomte & C. Retoré – « Towards a minimal logic for minimalist grammars : a transformational use of Lambek calculus », in *Formal Grammar, FG'99*, FoLLI, 1999, p. 83–92.
[42] ———, « Extending Lambek grammars : a logical account of minimalist grammars », in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL 2001* (Toulouse), ACL, July 2001, p. 354–361.
[43] D. Mazza – « Interaction nets : Semantics and concurrent extensions », Ph.D. Thesis, Université de la Méditerranée/Università degli Studi Roma Tre, 2006.
[44] S. McCall (éd.) – « Polish logic, 1920-1939 », Oxford University Press, 1967.
[45] N. Meggido – « Partial and complete cyclic orders », *Bulletin of the American Mathematical Society* **82** (1976), n° 2, p. 274–276.
[46] R. H. Möhring – « Computationally tractable classes of ordered sets », in *Algorithms and Order* (I. Rival, éd.), NATO ASI series C, vol. 255, Kluwer, 1989, p. 105–194.
[47] M. Moortgat – « Categorial type logic », in *Handbook of Logic and Language* [67], p. 93–177.
[48] R. Moot – « Automated extraction of type-logical supertags from the spoken dutch corpus », in *The Complexity of Lexical Descriptions and its Relevance to Natural Language Processing : A Supertagging Approach* (S. Bangalore & A. Joshi, éds), MIT Press, 2007.
[49] ———, « Filtering axiom links for proof nets », in *Proceedings of the 12th conference on Formal Grammar (FG 2007), Dublin, Ireland* (L. Kallmeyer, P. Monachesi, G. Penn & G. Satta, éds), CSLI Publications, 2007, To appear, ISSN 1935-1569.
[50] R. Muskens – « Lambdas, Language, and Logic », in *Resource Sensitivity in Binding and*

Anaphora (G.-J. Kruijff & R. Oehrle, éds), Studies in Linguistics and Philosophy, Kluwer, 2003, p. 23–54.
[51] M. Nivat – « Congruences de thue et t-langages », *Studia scientiarum mathematicarum hungarica* **6** (1971), p. 243–249.
[52] M. Pentus – « Lambek grammars are context-free », in *Logic in Computer Science*, IEEE Computer Society Press, 1993.
[53] ———, « Lambek calculus is np-complete », *Theor. Comput. Sci.* **357** (2006), n° 1-3, p. 186–201.
[54] G. Perrier – « Les grammaires d'interaction », Mémoire d'habilitation à diriger des recherches, Université Nancy 2, Janvier 2003.
[55] C. Pollard – « Higher-order categorical grammars », in *Categorial grammars – an efficient tool for natural language processing* (Montpellier), C.N.R.S., June 2004, p. 340–361.
[56] S. Pogodalla & C. Retoré – « Handsome non-commutative proof-nets : perfect matchings, series-parallel orders and hamiltonian circuits », Tech. Report RR-5409, INRIA, 2004, Presented at Categorial Grammars 2004, to appear in the Journal of Applied Logic.
[57] J. Pustejovsky – *The generative lexicon*, M.I.T. Press, 1995.
[58] C. Retoré – « Handsome proof-nets : perfect matchings and cographs », *Theoretical Computer Science* **294** (2003), n° 3, p. 473–488, Complete version RR-3652 `http://www.inria.fr`.
[59] C. Retoré – « The logic of categorial grammars – lecture notes », Tech. Report RR-5703, INRIA, 2005, ESSLLI 2000 / ACL 2001 / Bordeaux Master in CS Lecture Notes.
[60] ———, « Les mathématiques de la linquistique computationnelle. premier volet : la théorie des langages », *Gazette des Mathématiciens* (2008), p. 35–62.
[61] C. Retoré & S. Salvati – « Non-associative categorial grammars and abstract categorial grammars », in *New Directions in Type Theoretic Grammars* (R. Muskens, éd.), ESSLLI07, FoLLI, 2007, p. 51–58.
[62] D. Roorda – « Resource logic : proof theoretical investigations », Thèse, FWI, Universiteit van Amsterdam, 1991.
[63] ———, « Interpolation in fragments of classical linear logic », *Journal of Symbolic Logic* **59** (1994), n° 2, p. 419–444.
[64] E. Stabler – « Derivational minimalism », in *Logical Aspects of Computational Linguistics, LACL'96* (C. Retoré, éd.), LNCS/LNAI, vol. 1328, Springer-Verlag, 1997, p. 68–95.
[65] R. Thomason (éd.) – *The collected papers of richard montague*, Yale University Press, 1974.
[66] H.-J. Tiede – « Lambek calculus proofs and tree automata », in *Logical Aspects of Computational Linguistics, LACL'98, selected papers* (M. Moortgat, éd.), LNCS/LNAI, n° 2014, Springer-Verlag, 2001.
[67] J. van Benthem & A. ter Meulen (éds) – *Handbook of logic and language*, North-Holland Elsevier, Amsterdam, 1997.

10. Rectificatif

Correctif à l'article « Les mathématiques de la linguistique computationnelle. Premier volet : la théorie des langages » paru dans la *Gazette* 115.

Dans l'exemple de transducteur réalisant l'accord de l'adverbe *tout* exprimant le degré complet, j'ai commis de nombreuses inversions entre les mots *consonne* et *voyelle* ainsi qu'entre les règles à appliquer dans chacun des deux cas. Voici la figure et le texte que j'aurais dû mettre à la fin du paragraphe 2.3 pages 45-46.

Nous donnons en figure 5 un transducteur qui représente la règle d'accord de *tout* utilisé comme adverbe de degré. F et M désignent respectivement les traits *féminin* et *masculin*, S et P désignent respectivement les traits *singulier* et *pluriel*. Une transition K/es_K est une abréviation pour les transitions de mêmes états initial et final b/b, c/c ... (pour toute consonne, y compris le *h* aspiré), de

même V/V est une abréviation pour les transitions a/a, e/e (pour toute voyelle, y compris le *h* non aspiré). La transition $?/?$ récrit n'importe quel caractère en lui même. Le transducteur réalise entre autres *tout FP_entières* → *tout entières* et *tout FP_grandes* → *toutes grandes*.

La règle dit que *tout* est invariable, sauf si l'adjectif qui le suit est au féminin et commence par une consonne : *les fenêtres tout entières ouvertes* et *les fenêtres toutes grandes ouvertes* – voir par exemple le Grévisse, §955.

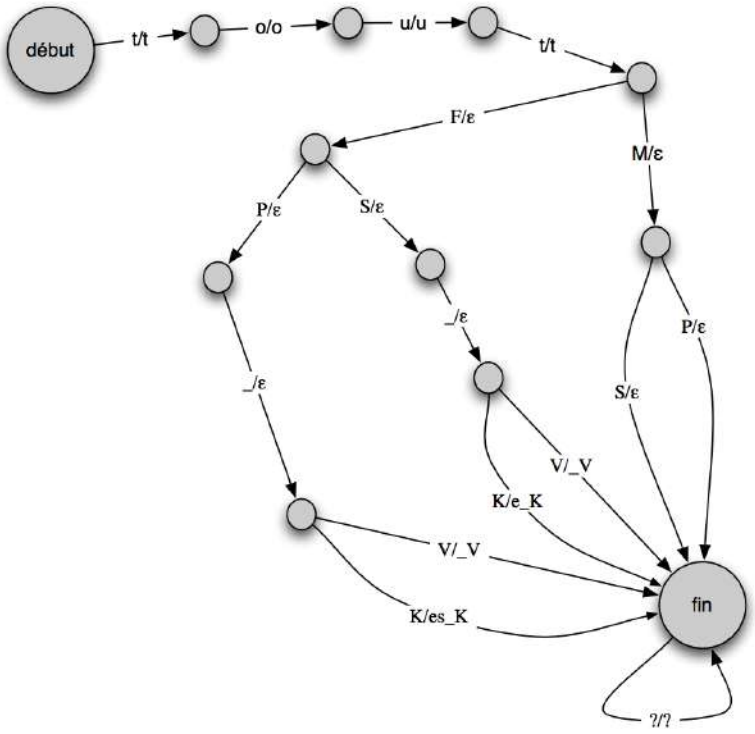

FIG. 5. Un tranducteur réalisant l'accord de l'adverbe *tout* exprimant le degré complet.