Verbum XL, 2018, n° 2, 143-172

BRÈVE GÉNÉALOGIE DES GRAMMAIRES CATÉGORIELLES

Jean-Pierre DESCLÉS
(STIH) Sens Textes Informatique Histoire
Sorbonne Université

RÉSUMÉ

Les Grammaires Catégorielles (GC) trouvent un ancrage philosophique et sémantique dans les travaux de l'École polonaise (Leśniewski, Ajdukiewicz) puis dans l'analyse syntaxique des langues naturelles de Bar-Hillel. Avec Lambek, on assiste à une interaction assumée entre une analyse syntaxique des langues et une approche algébrique de ces analyses. Autour des années 1990, les GC étendues s'appuient sur les propriétés du Calcul de Lambek pour augmenter les capacités d'analyse syntaxiques des langues. Les catégories syntaxiques sont alors analysées comme des types fonctionnels de la théorie des types de Church. Comme certaines analyses syntaxiques nécessitent des compositions et des transformations des types, il s'est établi des liens formels pertinents entre les formalismes des GC et de la Logique Combinatoire de Curry. Ces liens sont étudiés et exploités dans les modèles de la Grammaire Catégorielle Combinatoire (GCC) de Steedman, de la Grammaire Applicative Universelle (GAU) de Shaumyan et de la Grammaire Catégorielle Combinatoire Applicative (GCCA) de Biskri et Desclés.

ABSTRACT

Categorial grammars (GC) of languages have a philosophical and semantic anchoring in the works of the Polish School (Leśniewski, Ajdukiewicz) and in the syntactic analysis of natural languages by Bar-Hillel. Lambek studied the deep interactions between the syntactic analyses of natural languages and an algebraic approach of these analyses. Around 1990, the extended GC, linked to the mathematical properties of the Lambek calculus, increase the syntactic capacity of natural languages. In the formalism of CG, syntactic categories are analyzed as functional types of the Church's type theory. In syntactic processes, it is necessary to compose or to transform types to achieve a correct analysis of some sentences. By this way, it is established pertinent relations between the formalisms of CG and Curry's Combinatory Logic. These links are studied inside of the models of Categorial Combinatorial Grammar (CCG) of Steedman, Universal Applicative Grammar (GAU) of Shaumyan and Categorial Combinatorial Applicative Grammar (GCCA) of Biskri and Desclés.

INTRODUCTION

Les Grammaires Catégorielles (voir par exemple Gardies (1975), Bach (1988), Casadio (1988), Bourdeau (2002), Joray et Godart-Wendling (2002), Rivenc et Sandu (2005), Desclés *et al*. (2016 b : 305-320)) se positionnent à un carrefour important où linguistique (syntaxe et sémantique), philosophie, logique, mathématiques et, maintenant, informatique, convergent dans l'étude de certaines de leurs préoccupations respectives. Après les analyses logiques de Frege ([1893], 1967 ; 1971) puis les études sémantiques des langages logiques et philosophiques menées par l'Ecole polonaise (Leśniewski, 1929) autour des années 1900-1930, un formalisme, appelé Grammaire Catégorielle (désormais GC), est apparu, dans la période 1930-1960. Ce formalisme s'est focalisé sur la syntaxe des langues pour constituer le formalisme AB d'Ajdukiewicz (1935) et de Bar-Hillel (1950, 1953) ; il a fait apparaître une analogie formelle entre les structures syntaxiques des langues et les structures arithmétiques des rapports rationnels entre nombres entiers. La montée en puissance de la Grammaire Générative a contribué à « étouffer » ce genre d'étude de la linguistique mathématique, d'autant plus qu'il a été démontré que le modèle AB des GC avait seulement la complexité des « langages non contextuels » (*context free languages*) de la hiérarchie de complexité syntaxique définie par N. Chomsky, qui argumentait par ailleurs que les langues naturelles avaient une complexité syntaxique supérieure[1]. Autour de 1987-1989, il y a eu un réveil du courant des GC, avec les rencontres et l'école d'été tenues à Tucson (en Arizona), concrétisées par les contributions publiées dans l'ouvrage collectif *Categorial Grammars and Natural Language Structures* (Oehrle *et al*., 1988). On y constate que le modèle initial AB s'est progressivement complexifié en s'orientant vers un système algébrique axiomatisé, dit Calcul de Lambek (1958, 1961), et vers des Grammaires Catégorielles étendues, développées par différents auteurs (J. van Benthem, J. Lambek, A. Leconte, R. Montague, M. Moortgat, G. Morrill, C. Retoré...) qui exploitent les rapports entre les formalismes catégoriels et la théorie des types fonctionnels de Church (1941), les mécanismes de déduction naturelle (de Gentzen)[2], la théorie mathématique des catégories issue des premiers travaux d'Eilenberg et Mac Lane (1945)[3], la

---

1 Il s'agit de la hiérarchie entre langages formels : langages réguliers reconnus par des systèmes markoviens et automates à états finis, langages non contextuels reconnus par des automates à mémoires structurées en piles, langages contextuels et langages récursifs, voir par exemple Gross et Lentin (1967). Dans Buszkowski (1988), il est démontré que les GC simples (AB) ont seulement la complexité syntaxique des langages non contextuels (contexte free).

2 Sur une présentation de la « déduction naturelle » appliquée à l'analyse du langage, voir Desclés *et al*. (2010).

3 Sur la théorie mathématique des catégories, voir Mac Lane (1971) et Lawvere et Schanuel (2009).

théorie de la démonstration, les approches multimodales et la logique linéaire[4]... De son côté, Curry (1958) a développé un formalisme logique général, appelé Logique Combinatoire (LC), que l'on doit penser comme une logique d'opérateurs quelconques appliqués à des opérandes, ces opérateurs pouvant être composés ou transformés par des opérateurs abstraits (appelés combinateurs) (Curry et Feys 1958, Hindley et Seldin 2008, Desclés *et al.* 2016a, 2016b, 159-215). Cette logique d'opérateurs (dont la logique classique des prédicats devient un cas particulier) peut être restreinte par une prise en compte de différents types fonctionnels (d'opérateurs et d'opérandes), déjà utilisés par Curry (1934), dans son analyse des catégories grammaticales qui associe étroitement les types fonctionnels aux catégories syntaxiques des langues, les unités linguistiques étant des opérateurs ou des opérandes absolus de ces catégories (voir Curry 1961, Curry *et al.* 1958 : 274-276)[5]. En 1965, le linguiste S. Shaumyan (1977, 1987), qui travaillait à l'époque à Moscou, s'est inspiré des types fonctionnels de H. Curry pour son étude des catégories des systèmes sémiotiques généraux, analysés comme des systèmes d'opérateurs appliqués à des opérandes ; il a également su utiliser, à bon escient, les combinateurs pour mener des études grammaticales plus poussées dans le cadre d'une Grammaire Applicative Universelle (GAU) qui décrit formellement un « langage génotype » qui se projette, à l'aide de combinateurs, dans les structures syntaxiques plus spécifiques des langues, dites « langues phénotypes ». Indépendamment de tous ces travaux sur les GC et les types fonctionnels, Harris (1976, 1982) entreprend de décrire, à partir de 1976, une « Grammaire d'opérateurs » des langues, celle-ci est, après examen, une nouvelle forme (implicite) de GC, accompagnée de processus de réductions des phrases à des « phrases sources », ces processus pouvant être, sans doute, formalisés à l'aide des combinateurs de Curry. Un lien entre les GC étendues et des combinateurs a été établi par Steedman (1988, 1989, 2000) dans ce qu'il a appelé *Combinatory Grammars*. Ce lien a été systématiquement développé par Biskri (1995) et Desclés (Biskri et Desclés, 1996, 1997) dans la Grammaire Catégorielle Combinatoire et Applicative (GCCA). Dans ce dernier formalisme, les compositions entre types (ou catégories) et les changements de types, nécessaires à de nombreuses analyses syntaxiques des langues, sont directement associés aux rôles tenus par les combinateurs dans la Logique Combinatoire Typée (LCT), conduisant à une analyse applicative des structurations (morpho-syntaxiques et sémantiques) des systèmes sémiotiques que sont les langues naturelles. Nous allons développer quelques uns des points qui viennent d'être évoqués.

---

4 Voir, par exemple, un usage de la logique linéaire (Girard, 1987) qui est intégrée au formalisme de la GC dans Casadio (2002).

5 Sur l'importance de la notion d'opérateur en linguistique, voir Desclés (2009).

1. CATÉGORÈMES ET SYNCATÉGORÈMES

Dans son approche mathématique de la logique, qui ainsi se libérait des carcans de la logique aristotélicienne, Frege (1891, 1893) a distingué deux types d'unités, les unités non saturées ou incomplètes et les unités saturées ou complètes. Les unités non saturées sont des fonctions (ou des opérateurs) qui nécessitent d'autres unités (des arguments ou opérandes) pour construire des résultats :

> Ici, nous arrivons à ce qui distingue les fonctions des nombres. [...] le signe d'une fonction est insaturé (*ungesättigt*), il doit être complété avec un signe numérique, que nous appelons alors le signe de l'argument. [...] Les signe de fonction ne peuvent apparaître seuls, comme le font les signes numériques, [...] mais seulement complétés par un signe qui désigne (*bezeichnet*) ou indique (*andeutet*) un nombre. (Frege, *Was ist eine Funktion*, 1904)

L'écriture traditionnelle '$y = 2*x^2 + 3$' d'une fonction est ambiguë puisqu'elle désigne plutôt le résultat 'y' d'une fonction pour l'argument 'x' et non la fonction « en soi ». La fonction verbalisée par « deux fois le carré d'un nombre auquel on additionne 3 » pourrait être exprimée par l'écriture incomplète '$2*( )^2 + 3$' ; celle-ci permet de déterminer une valeur précise lorsqu'elle est complétée par un nombre déterminé (par exemple par '5', d'où la valeur '53' de la fonction), ou, plus généralement, par une variable 'x' (un nombre quelconque) pour représenter un résultat dépendant de cette variable. La λ-notation introduite par A. Church dans le λ-calcul (Church 1941, Desclés *et al*., 2016a : 145-152, 2016b : 55-66), lève l'ambiguïté de l'écriture 'y = 2*x2 + 3' en faisant appel à une « variable liée » 'ξ', pour noter la fonction par '$\lambda\xi.\ [2*\xi^2 + 3]$' ; lorsque cette fonction est appliquée à un argument, par l'opération d'application, notée '@', la valeur de la fonction est construite en substituant l'argument (un numérique, ou une variable), à l'occurrence 'ξ', liée par $\lambda\xi$, dans l'expression de la fonction ; par exemple :

$$\lambda\xi.\ [2*\xi^2 + 3]\ @\ 5 \Rightarrow 2*(5)^2 + 3 = 2*(25) + 3 = 50 + 3 = 53$$
$$\lambda\xi.\ [2*\xi^2 + 3]\ @\ x \Rightarrow 2*(x)^2 + 3 = y$$

Citons la définition de la fonction donnée par Church :

> A function is a rule of correspondence by which when anything is given (as argument) another thing (the value of the function for that argument) may be obtained. That is, a function is an operation which may be applied on one thing (the argument) to yield another thing (the value of the function)... (Church 1941).

Avant Church et le λ-calcul, Frege (1893) avait déjà introduit des notations analogues en étendant le concept de la « fonction » numérique au domaine des entités symboliques afin de mathématiser la logique des prédicats et de la quantification. Pour Frege, un concept (ou un prédicat unaire) prend

la forme d'une fonction ayant pour valeur l'une des deux valeurs de vérité de {Vrai, Faux} ; par exemple, le concept « être un nombre » est tel que :

$\lambda\xi$ [ξ est-un-nombre] @ 5 $\Rightarrow$ Vrai
$\lambda\xi$ [ξ est-un-nombre] @ a $\Rightarrow$ Faux

La distinction frégéenne « fonctions / arguments », fondamentale en mathématiques, est également présente, sous une autre forme, dans les *Recherches Logiques* du philosophe Husserl (1913) qui reprend l'opposition traditionnelle entre expressions catégorématiques et expressions syncatégorématiques ; les premières, ayant une signification complète, sont autonomes, les secondes ont une signification incomplète car leurs significations ne se déterminent que conjointement avec d'autres expressions (voir Rivenc et Sandu, 2009 : 11-24) ; alors que 'Luc' est un catégorème qui réfère directement à une entité externe, un verbe comme 'grandit' est un syncatégorème, avec une signification incomplète ; mais, en le composant avec un catégorème comme 'Luc', il construit la proposition '(Luc) grandit' qui est une expression logico-linguistique autonome. Il est légitime de rechercher les lois de construction des significations par des combinaisons (possibles ou impossibles) de syncatégorèmes avec des catégorèmes. C'est à la « Grammaire pure »[6] qu'il convient de répartir les expressions du langage en catégories puis à établir les lois qui régissent les agencements d'une expression d'une des ces catégories avec les expressions de telle ou telle autre catégorie :

> De là découle la tâche importante, également fondamentale pour la logique et pour la grammaire, de mettre en évidence cette organisation a priori, qui s'étend à tout le domaine des significations, et d'explorer dans une « morphologie des significations », le système apriorique des structures formelles, c'est-à-dire de celles qui laissent de côté toutes les particularités concrètes des significations. (Husserl, *Recherches logiques*, IV, 115)

Pour le logicien polonais Leśniewski (1929), les syncatégorèmes fonctionnent comme des « foncteurs » en reprenant la terminologie de l'Ecole de Varsovie :

> Bien que ma conception des catégories sémantiques, quant à ses conséquences théoriques, soit en étroite conformité formelle avec les célèbres « Théories des types logiques », elle se rattache plutôt, en ce qui concerne son sens intuitif, à la tradition des catégories d'Aristote, des parties du discours de la grammaire traditionnelle ou des *Bedeuntungskategorien* d'Edmund Husserl. (Leśniewski 1929).

Les différentes catégories se ramènent finalement à la distinction « fonction / argument » de Frege. Un foncteur (un verbe, un adjectif, un connec-

6 L'idée de « grammaire pure » a été introduite dans Husserl ([1913], 1972 : 85-138) ; voir aussi Gardies (1975).

teur...) se comporte comme une fonction qui construit une nouvelle expression à partir d'un argument donné. Par exemple, le verbe conjugué 'grandit' est analysé comme la fonction 'λξ.[grandit (ξ)]' qui construit une proposition qui dénote la valeur de vérité « vraie » ou « fausse » selon que son argument grandit effectivement ou ne grandit pas ; le foncteur adverbial 'vite', appliqué au syncatégorème 'grandit' construit le syncatégorème 'grandit vite' qui, en s'appliquant à son tour au catégorème 'Luc', construit l'expression propositionnelle 'grandit vite (Luc)'.

2. GRAMMAIRES CATÉGORIELLES SIMPLES (AB)

Ajdukiewicz (1935) a repris les idées de Leśniewski avec cependant une visée nettement plus syntaxique. Ajdukiewicz retient deux catégories de base 's' (pour sentence) et 'n' (pour entité nominale) ; les catégories dérivées sont toutes de la forme : 'C/ C_1...C_n' dans laquelle 'C' et 'C_1', ..., 'C_n' sont des catégories de base ou des catégories déjà dérivées ; un élément (ou une instance) d'une telle catégorie dérivée est un foncteur qui, en s'appliquant à des arguments de catégories 'C_1', ..., 'C_n' construit un élément de catégorie 'C'. Ainsi, les verbes 'marche' et 'admire' sont des foncteurs de catégories respectives 's/n' et 's/nn' ; l'adjectif 'grand' est un foncteur de catégorie 'n/n' qui, après son application à 'homme', de catégorie 'n', construit le nominal 'grand homme' de catégorie 'n' ; l'adverbe 'vite' est un foncteur de catégorie '(s/n)/(s/n')), qui s'applique à 'marche', de catégorie 's/n', afin de construire le foncteur 'marche vite' de catégorie 's/n', qui, à son tour s'applique à 'Luc', de catégorie 'n', d'où la proposition '(Luc) marche vite', de catégorie 's'. Ajdukiewicz fait usage d'une notation fractionnaire 'x/y' composable avec la simple concaténation des catégories, d'où l'analogie formelle avec la simplification arithmétique des fractions :

(a)	(2/3)*3 = 2 = 3*(2/3)	(a')	(p/q)*q = p = q*(p/q)
(b)	(s/n) n = s = n (s/n)	(b')	(x/y) y = x = y (x/y)

La simplification des catégories concaténée dans (b) et sa généralisation (b') sont analogues à la simplification (a) des fractions en arithmétique et (a') en algèbre. L'analogie formelle prend sens lorsque la notation 's/n' (plus généralement 'x/y') est interprétée comme étant le type fonctionnel 'x/y' d'un opérateur (ou foncteur) qui s'applique à une expression de type 'y' pour former une nouvelle expression de type 'x' (« *A quasi-arithmetical notation for syntactic description* », selon Bar-Hillel). Assigner le type 'x' à une entité 'X' (qualifiée d'instance de 'x'), ce que l'on note [X : x], signifie que 'X' appartient à la catégorie de type 'x', en possédant les mêmes propriétés essentielles que les autres membres de la catégorie 'X' et en étant l'objet des mêmes opérations de composition ou de non composition avec les entités de types différents . Dans une GC, les entités d'un même type sont soit des opérandes absolus s'il s'agit d'un type de base, soit des opérateurs susceptibles

de s'appliquer à des opérandes d'un certain type pour construire des expressions, dites applicatives, d'un autre type, ces expressions applicatives pouvant fonctionner, à leur tour comme des opérateurs ou comme des opérandes d'autres opérateurs. La notion générale de type est opératoire en linguistique (avec les types syntaxiques), mais également en logique (types des entités individuelles, des prédicats, des connecteurs, des quantificateurs, des opérateurs modaux...), ou en informatique (différents types des expressions constitutives d'un programmes dans les divers langages de programmation) ; voir Bourdeau (2002), Desclés *et al*. (2016 b : 305-355).

Bar-Hillel (1953) a cherché à concilier l'ordre syntagmatique des expressions linguistiques dans les langues avec une organisation applicative structuré par les opérations d'application d'opérateurs à leurs opérandes. Pour cela, il oriente les types en distinguant le type 'x/y' du type 'y\x' d'un opérateur, selon que l'opérande, de type 'y', est positionnée sur la chaîne syntagmatique à droite, ou à gauche, de l'opérateur ; en notant par '+' la concaténation syntagmatique, les deux règles d'application à droite, respectivement à gauche, sont formulées comme suit :

```
[X :x/y]  +  [Y :y]                [Y :y]  +  [X :y\x]
---------------------->            ----------------------<
      [XY :x]                            [YX :x]
```

Les types de base sont désignés par 'S' et 'N' ; les verbes comme 'marche' et 'admire' ont pour types (syntaxiques) assignés : [marche : N\S] et [admire : (N\S)/N] ; on en déduit les deux constructions applicatives obtenues par les règles d'application à droite et à gauche :

```
[Luc :N]+[marche :N\S]           [Luc :N]+[admire :(N\S)/N]+[Marie :N]
-----------------------------<             ------------------------------------>
[Luc marche : S]                           [admire Marie : N\S]
                                 ----------------------------------------------<
                                           [Luc admire Marie : S]
```

L'analyse syntaxique de la séquence 'Luc court vite vers son école' s'effectue en plusieurs étapes, en posant que le type syntaxique de l'adverbe 'vite' est 'α\α' avec [α = N\S] et celui de la préposition 'vers' est '(α\α)/N' :

```
[Luc :N]+[court :N\S]+[vite :α\α]+[vers :(α\α)/N] +[son :N/N]+[ école :N]
            -------------------------<                 -------------------------->
                  α  =  N\S                                         N
                                        --------------------------------------------->
                                                                   α\α
            ---------------------------------------------------------------------------<
                                           α  =  N\S
----------------------------------------------------------------------------------------<
                                            S
```

Le calcul de la correction syntaxique s'effectue uniquement sur les séquences des types syntaxiques, indépendamment des instances (et donc de leurs significations) ; cela conduit à engendrer (par les règles d'application) des relations de réduction entre types ; pour les exemples précédent, nous obtenons :

$N + N\backslash S \rightarrow S$
$N + (N\backslash S)/N + N \rightarrow N + N\backslash S \rightarrow S$
$N + N\backslash S + x\backslash x + (x\backslash x)/N + N/N + N \rightarrow \ldots \rightarrow N + N\backslash S \rightarrow S$

Avec les types syntaxiques orientés, Bar-Hillel (1953) unifie l'approche du logicien polonais Ajdukiewicz (1935) avec les analyses distributionnelles du linguiste Harris (1951) qui, par son approche structurale, a dégagé des structurations mathématiques sous forme de classes d'équivalence d'unités linguistiques et de relations entre ces classes.

3. TYPES FONCTIONNELS DE CHURCH

Comme l'a fait observer B. Russell à G. Frege, qui venait de publier son système définitif, la construction de certaines expressions engendrées sans aucune restriction, par des applications de fonctions à des arguments, peut conduire à des contradictions logiques, les fameux paradoxes que la logique doit éviter. Considérons par exemple la propriété 'F' qui est, par définition « la propriété de non applicativité d'une propriété 'f' quelconque », c'est-à-dire telle que $[F(f) =_{def} N(ff)]$ (où N désigne la négation propositionnelle), ou encore la classe C telle que [C =$_{def}$ la classe des objets qui ne s'appartiennent pas à eux-mêmes]. La propriété F et la classe C, ainsi définies, aboutissent à former les propositions contradictoires : $[FF \Leftrightarrow N(FF)]$ ou $[C \in C \Leftrightarrow C \notin C]$[7]. Il est donc nécessaire d'apporter certaines restrictions dans la formation des expressions pour bloquer les constructions indésirables. Les théories des types logiques répondent à cet objectif, elles sont destinées à assigner des types aux expressions et à se donner des règles de composition sur les types des expressions composées de façon à pouvoir éliminer les expressions « mal typées » qui ne respecteraient pas les règles sur les types : seront non paradoxales toutes les expressions qui, par leurs constructions, respectent les contraintes restrictives apportées par les types de leurs constituants. Church (1940) a défini un système de typage fonctionnel des expressions logiques, distinct des différents systèmes de typage proposés par Russell (1905)[8] . Church considère des types (ou sortes) de base et dérivent tous les types fonctionnels à partir de ces types de base par les règles suivantes :

(F_1) les types de base sont des types fonctionnels ;

---

7 Voir une présentation détaillée dans Desclés *et al.* (2016a : 187-199).

8 Voir Desclés *et al*. (2016a : 219-234).

(**F₂**) si α et β sont des types fonctionnels alors **F**αβ est un type fonctionnel.

Le type fonctionnel **F**αβ est le type des opérateurs pouvant s'appliquer à des expressions de type α pour construire des résultats de type β ; l'assignation d'un type fonctionnel **F**αβ à une expression applicative X est notée [X : **F**αβ] ; le résultat de l'application de X à une expression applicative Y de type β, est noté 'XY', dont le type est : [XY : β]. En nous restreignant à des expressions applicatives (composées uniquement d'opérateurs appliqués à des opérandes) fonctionnellement bien typées, on peut ainsi éviter de construire les expressions paradoxales découvertes par Russell, et à sa suite, par d'autres mathématiciens. Le type fonctionnel '**F**αβ' de Church correspond exactement à la catégorie désignée par la notation infixée 'β/α' d'Ajdukiewicz[9]. La règle déductive d'application est accompagnée de deux règles complémentaires :

[X :**F**αβ] [Y :α]	[X :**F**αβ] [XY :β]	[XY :β] [Y :α]
----------	----------	----------
[XY :β]	[Y :α]	[X :**F**αβ]

Dans nos analyses syntaxiques effectuées à l'aide des GC, nous avons introduit des variables de types, ce qui permet de mieux dégager le rôle multiple de certaines catégories grammaticales[10] comme celles :

- des opérateurs de détermination (adjectifs, adverbes,) de schéma de type **F**αα où α est le type N d'un nominal ou le type **F**NS d'un syntagme verbal ;
- des connecteurs (de phrases, noms, adjectifs...) de schéma de type **F**α (**F**αα) (avec [α = S] ou [α = N] ou [α = **F**NN]...) ;
- des opérateurs prépositionnels **F** N (**F** αα) (avec [α = **F** **F**NS **F**NS] ou [α = **F**NN]) ;
- des opérateurs de nominalisation, de schéma de type **F**αN, la nominalisation s'effectuant à partir d'un adjectif [α = **F**NN], d'un adverbe [α = **F** **F**NS **F**NS], d'un verbe [α = **F**NS], d'une phrase [α = S].

Dégager les différents rôles sémantiques de prédication, de détermination, de nominalisation et de « translation »[11] (ou de transposition) est fondamental pour l'analyse typologique puisque ces rôles peuvent recevoir des catégorisations et rôles syntaxiques différents selon les langues. Pour

---

9 Voir le tableau des principales catégories syntaxiques dans Desclés *et al*. (2016a : 223-225).

10 Voir Desclés *et al*. (2016a : 224-234).

11 La translation est prise au sens de Tesnière (1959) ; il s'agit d'un changement du type d'une unité linguistique sous la portée d'opérateurs spécifiques, par exemple le nominal '*Pierre*' de type N devient le déterminant adjectival '*de Pierre*' de type **F**NN, en tombant sous la portée de l'opérateur '*de*' de type **F**N**F**NN.

prendre un exemple, le marqueur verbal 'est', souvent appelé « copule » dans la tradition aristotélicienne, est un opérateur dont le type fonctionnel est polymorphe : il a pour schéma **F**α**F**NS, et est analysé comme « un constructeur de prédicat verbal unaire » de type **F**NS, la construction étant effectuée à partir d'un nom propre ou d'un syntagme complètement déterminé (avec [α = N]), dans *Napoléon est Bonaparte* / *Napoléon est l'empereur des français)*, d'un adjectif (avec [α = **F**NN] dans *Napoléon est prudent*) ou encore d'un adverbe (avec [α = **F** **F**NS **F**NS] dans *Napoléon est contre*) (voir Desclés *et al*. 2016b : 358, 461-466)... Il est bien connu que la polysémie du marqueur 'est' du français (et de ses « équivalents » dans d'autres langues) n'est pas universelle car les différentes valeurs qui lui sont attachées s'expriment souvent par différents marqueurs dans de nombreuses langues non indo-européennes[12].

4. ANALYSE SYNTAXIQUE ET LOGIQUE COMBINATOIRE

Reprenant une étude antérieure sur les types logiques[13], Curry (1961) l'applique systématiquement à l'analyse linguistique, avec les types de base N (pour nom) et S (pour phrase) et les types fonctionnels dérivés associés aux catégories grammaticales (verbes, adjectifs, adverbes, prépositions, articles ...). Dans le formalisme de la Logique Combinatoire (LC), que Curry a considérablement développé[14], tout opérateur est unaire avec le type fonctionnel **F**αβ, dans lequel β peut être le type d'un autre opérateur. Ainsi, le type d'un verbe transitif (comme 'mange') est **F**N **F**NS : en s'appliquant à un premier opérande, l'opérateur unaire 'mange' produit une expression verbale intransitive de type **F**NS, celle-ci peut s'appliquer à un opérande qui joue le rôle syntaxique d'un sujet[15], pour aboutir à la construction d'une phrase, de type S, et à sa présentation applicative préfixée :

---

12 Voir, entre autres, Benveniste (1966 : 187-207) et Culioli *et al*. (1981).

13 Curry (1934), Curry et Feys (1958 : 274-275).

14 Sur une présentation de la logique combinatoire, voir Curry et Feys (1958), Hindley et Seldin (2008), Ginisti (1997), Desclés *et al* (2016a) ; ce formalisme d'opérateurs s'est développé indépendamment de Schönfinkel (1924), selon le témoignage de Curry lui-même dans Curry et Feys (1958 : 184). Sur son interprétation philosophique et linguistique, voir Desclés *et al*. (2016b : 159-226).

15 Il apparaît ainsi que le « sujet syntaxique » est le dernier terme qui entre dans la construction prédicative. Certains sujets syntaxiques ne sont pas des sujets sémantiques ; par exemple dans *Il a été vendu beaucoup de cigarettes cette année dans ce pays*, le sujet syntaxique *il* n'est pas un sujet sémantique.

[Ève :N]+[mange :**F**NFNS]+[une :**F**NN]+[belle :**F**NN]+[pomme :N]
----------> [belle pomme :N]
----------> [une (belle pomme) :N]
----------> [mange (une (belle pomme)) :**F**NS]
<---------- [(mange (une (belle pomme)) Ève :S]

Pour Curry (1961 : 67) « we can profitably study grammatical structure as such, apart from its representation in terms of concatenation ». On obtient une équivalence entre le type **F**N**F**NS et le type noté '$\mathbf{F}_2$(NN)S' (notation qui rappelle celle d'Ajdukiewicz). Cette équivalence est appelée principe de curryfication[16], qui est totalement justifié lorsqu'on interprète, dans le cadre de la théorie des ensembles, un type fonctionnel **F**$\alpha\beta$ comme étant le type fonctionnel de l'ensemble, désigné par Ens(A,B), de toutes les fonctions de l'ensemble A (des entités de type α) dans l'ensemble B (des entités de type β). Dans la théorie des ensembles, pour trois ensembles A, B et C, nous avons les bijections suivantes entre les ensembles de fonctions :

$$\text{Ens (A x B, C)} \cong \text{Ens (A, Ens (B, C))} \cong \text{Ens (B, Ens (A, C))}$$

Ces bijections signifient que, à chaque fonction 'f_2' du produit cartésien A x B dans C, c'est-à-dire que $f_2 \in$ Ens (A x B, C), correspondent d'une part, une unique fonction '$\hat{}f_1$' de A dans l'ensemble Ens(B,C) des fonctions de B dans C et d'autre part, une unique fonction '$f_1\hat{}$' de B dans l'ensemble Ens(A,C) des fonctions de A dans C, telles que, lorsque $x \in A$, $y \in B$, $<x,y> \in A \times B$, on ait :

$$(\hat{}f_1\,(x))\,(y) = f_2\,(<x,y>) \in C \quad \text{et} \quad (f_1\hat{}\,(y))\,(x) = f_2\,(<x,y>) \in C$$

Avec le principe de curryfication, l'analyse prédicative d'une phrase devient plus subtile, puisqu'il faut devoir argumenter, sur le plan théorique de la linguistique, les raisons qui feraient préférer l'une des deux analyses : celle où le sujet est le premier terme de la construction prédicative, les autres termes étant des compléments[17], ou bien celle où le sujet est le dernier terme qui entre dans la construction prédicative. Pour la phrase *Luc admire sa femme*, exprimée selon l'ordre syntagmatique linéaire, nous avons les deux analyses prédicatives présentées par les expressions applicatives préfixées :

---

16 La curryfication avait déjà été formulée dans Schönfinkel (1924). Dans le formalisme de la Logique Combinatoire (LC), le principe de curryfication joue un rôle important puisque tous les opérateurs sont unaires, certains opérateurs construisent des opérateurs à partir d'opérandes.

17 C'est la position grammaticale classique de la suite des compléments ; pour Jespersen (1971), le sujet est un complément de « rang zéro », les autres compléments étant de rang 1, 2, 3...

(i) $(_2\ (_1$ admire (Luc) $_1)$ (sa femme) $_2)$
(ii) $(_2\ (_1$ admire (sa femme) $_1)$ (Luc) $_2)$

Cette discussion est effacée par la simple décomposition prédicative de la logique classique ou encore par la théorie des stemmas de Tesnière (1959), qui ne hiérarchisent pas les actants en relation de dépendance directe avec le prédicat[18], d'où la troisième analyse :

(iii) $(_1$ admire (<Luc, sa femme >) $_1)$

S'inspirant de Curry, Shaumyan (1965) a proposé une analyse sémiologique (morphologique et syntaxique) des phrases à l'aide d'épisémions et de sémions. Les épisémions sont en fait des types fonctionnels (de Church) et les sémions des instances de ces types[19] ; lorsqu'un sémion est une instance d'un épisémion du type **F**$\alpha\beta$, ce sémion fonctionne comme un opérateur qui s'applique à un sémion de type α pour construire un sémion de type β. Prenons deux exemples empruntés à la morphologie avec les types de base désignés par 's' et 'n'. Dans la décomposition du mot 'fill-ette', le suffixe '-ette' est un sémion de type **F**nn ; il s'applique au sémion 'fille' (de type n) pour former le mot 'fillette' (de type n). La décomposition du mot 'sur-pass-ait' considère le sémion préverbal (préfixé) 'sur-' comme un opérateur qui en s'appliquant au radical verbal 'pass-' construit le radical verbal dérivé 'surpass-' ; quant au sémion suffixal d'imparfait '-ait', il construit la forme verbale conjuguée 'surpassait'. Shaumyan (1977, 1987) a étendu ses analyses en ayant recours aux opérateurs abstraits (appelés combinateurs) de la Logique Combinatoire dans le cadre de sa Grammaire Applicative Universelle (GAU) où les phrases sont d'abord analysées morphologiquement et syntaxiquement (à laide d'un GC) puis ensuite reliées à d'autres phrases par des relations paraphrastiques[20] comme dans les exemples suivants :

Il y a un lièvre qui a été tué → On a tué un lièvre.
Luc se rase → Luc rase Luc.
Luc s'endort → Luc change lui-même d'état en passant de Luc ne dort pas à Luc dort.
Le poulet se mange facilement → Il est facile de manger du poulet.

Totalement indépendamment des GC et des types de Church, dans une approche moins strictement structuraliste que celle de ses recherches antérieures, Harris (1976, 1982) a développé une Grammaire d'opérateurs dans laquelle il retrouve les types fonctionnels, avec cependant une notation différente de celle qui est en usage dans le GC. Les types de base sont 'n' (nom)

---

18 Plusieurs arguments linguistiques conduisent à adopter plutôt l'analyse et la représentation (ii), dans laquelle le « sujet syntaxique », obligatoire pour que la prédication soit complète, est le dernier des arguments qui entre dans la construction prédicative.

19 Voir Guentchéva (1976).

20 Voir Shaumyan (1977, 1987) ; Desclés, Guentchéva et Shaumyan (1985, 1986).

et les discours ; les types fonctionnels d'un verbe intransitif, transitifs, à trois actants, sont des types d'opérateurs notés respectivement 'On', 'Onn', 'Onnn'. Harris convient qu'il s'agit d'une forme de GC :

> La signification des mots [...], sont en partie déterminés à partir de leurs combinaisons plutôt qu'à partir de leur stricte identité [...] l'importance de cette relation deviendra plus claire lorsque l'on verra que toutes les classes ultérieures, toutes les opérations dans la langue et presque toutes les significations, sont formulées à partir de constructions résultant de cette relation. Il faudrait noter que la relation opérateur-argument produite par cette dépendance présente des similarités importantes avec les foncteurs de la grammaire catégorielle en logique. Les différences viennent de ce que la syntaxe de la logique et la syntaxe des langues naturelles n'ont pas la même finalité. (Harris, *La langue et l'information* (1988/2007 : 37).

Harris entreprend également des « réductions » de phrases à leurs « sources » interprétatives. Par exemple, *Il arrivera* est réduite, en plusieurs étapes, à la source formulée par une glose du genre : « Je dis que son arrivée a lieu à un moment qui est un instant après l'instant que je dis » (Harris, 1976 : 164)[21].

5. CALCUL DE LAMBEK ET GRAMMAIRES CATÉGORIELLES ÉTENDUES

Lambek (1958, 1961) approfondit l'analogie entre les formalismes syntaxiques des GC et le calcul algébrique en axiomatisant le calcul sur les types syntaxiques des langues sous la forme d'un calcul déductif présenté dans le style de la « déduction naturelle » de Gentzen[22]. Les règles sur les types de ce qui est devenu le « calcul de Lambek ». Nous n'allons pas présenter systématiquement ces règles[23]. L'interprétation intuitive des règles de ce calcul est directement donnée par la compréhension de l'opération d'application d'un opérateur à un opérande de type compatible avec le type de l'opérateur, notion qui est à la base des GC. Donnons deux exemples de ces règles de déduction entre des relations entre types, présentée avec une seule orientation :

$$[x*y \rightarrow z] \Rightarrow [x \rightarrow z/y] \quad \text{et} \quad [x \rightarrow z/y] \Rightarrow [x*y \rightarrow z]$$

Ces règles sont analogues aux règles de simplification des nombres fractionnaires :

$$[p * q = r] \Rightarrow [p = r/q] \quad \text{et} \quad [p = r/q] \Rightarrow [p*q = r]$$

---

21 Nous avons montré que cette réduction harissienne peut être adéquatement formalisée en faisant appel à des opérateurs de composition de la logique combinatoire ; voir la réduction de *il arrivera* dans Desclés (2016 : 104).

22 Pour une présentation de la logique classique par la « déduction naturelle », voir Desclés *et al*. (2010).

23 Voir par exemple Desclés *et al*. (2016b : 321-339).

Ce genre de calcul permet de démontrer des théorèmes sur les relations entre types qui trouvent également une compréhension directe lorsque les types sont interprétés comme des types d'opérateurs. Donnons des exemples de théorèmes qui établissent des relations entre types :

Application :	(z/y)*y → z
Composition :	(z/y) *(y/x) → z/x
Division :	(z/y) → (z/x)*(y/x)
Curryfication :	z/yx → (z/y)/x
Montée de type :	z → z/(z\y)

La première relation de simplification traduit directement l'opération d'application d'un opérateur d'un certain type sur un opérande d'un type compatible avec celui de l'opérateur; la seconde formalise une composition entre les types fonctionnels ; la troisième établit une division d'un type fonctionnel ; la quatrième exprime le principe de curryfication ; la cinquième introduit un changement de type (que nous allons commenter plus loin). Le calcul de Lambek permet de capter certaines analyses syntaxiques (comme le traitement des problèmes de coordination) que le simple calcul catégoriel AB ne permet pas. Par ailleurs, il a été démontré que le calcul fonctionnel des types de Lambek est formellement analogue au calcul (intuitionniste) des propositions sans négation. En effet, la relation (ou simplification entre types) '(x/y) * y → x' est formellement analogue à la règle (non orientée) de déduction, dite 'modus ponens' du calcul des propositions (p et q étant des propositions quelconques), c'est-à-dire :

Calcul de Lambek (application)	Calcul des propositions (modus ponens)
(x/y)*y → x	(q ⇐ p) & p ⊢ q

Par ailleurs, il a été démontré que le Calcul de Lambek est décidable, ce qui est un apport considérable pour entreprendre une automatisation de l'analyse syntaxique effectuée dans ce cadre formel. J. Lambek a étudié également les rapports pertinents entre d'un côté, le calcul sur les types fonctionnels, désigné par « calcul catégoriel », et d'un autre côté, « la théorie des catégories cartésiennes fermée » étudiée par Lawvere dans l'approche « catégorique » des mathématiques de la « théorie des catégories »[24]. Dans cette théorie mathématique, les opérateurs de type $\underline{F}\alpha\beta$ (noté aussi 'β/α' lorsque l'on tient compte de l'orientation) sont représentés par des flèches entre le type α et β ; les propriétés des opérateurs sont alors exprimées au moyen de diagrammes commutatifs. Ces analogies formelles entre les types syntaxiques, le calcul algébrique de Lambek et certains diagrammes commutatifs d'une théorie cartésienne fermée[25], témoignent que la syntaxe des

[24] Il s'agit d'être sensible à l'opposition entre « categorical » (catégorique) et « categorial » (catégoriel), signalée par J. Lambek dans Godart-Wendling (éd.) (2002 : 68).

[25] Voir, par exemple Lambek (1980) et Lawvere et Schanuel (2009).

langues naturelles a une structuration mathématique sous-jacente qu'il convient de révéler et d'approfondir.

Moague (1974) a étudié les rapports entre la théorie de la quantification opérée par la logique classique (calcul des prédicats) et celle les syntagmes quantifiés appréhendés par la syntaxe des langues naturelles. Il s'agit de comparer les analyses des sujets syntaxiques dans : (a) *Tous sont mortels*. / (b) *Luc est mortel*. / (c) *Tous les hommes sont mortels*. A la suite des analyses de Frege (1893), le quantificateur 'Tous' dans (a) est analysé sous la forme d'un opérateur, noté, en utilisant une variable liée, par '∀x' ; cet opérateur s'applique au prédicat verbal 'être-mortel' pour construire la proposition logique exprimée '(∀x) [être-mortel (x)]'. En se situant dans le cadre catégoriel des types syntaxiques fonctionnels (qu'il redéfinit indépendamment d'Ajdukiewicz, Bar-Hillel, Curry et Lambek), R. Montague assigne le type syntaxique 'S/(N\S)' au sujet syntaxique 'Tous', d'où la relation qui construit une expression applicative :

(a') [Tous :S/(N\S)]+[être-mortel :N\S] → [tous (être-mortel) :S]

Dans l'analyse catégorielle de (b), le sujet syntaxique 'Luc', de type 'N', fonctionne comme un opérande du prédicat verbal 'être-mortel', de type 'N\S' :

(b') [Luc :N]+[être-mortel :N\S] → [être-mortel (Luc) :S]

La comparaison des deux relations (a') et (b') montre immédiatement que sujets syntaxiques 'Tous' et 'Luc' ne se voient pas assigner le même rôle et le même type syntaxique dans la prédication puisque, dans (a'), le sujet est un opérateur appliqué au prédicat verbal tandis que, dans (b'), le sujet est un opérande du prédicat verbal. Afin d'unifier les deux analyses, Montague introduit une règle de « montée de type » (dite type raising), exprimée par la relation entre types 'N → S/(N\S)'. Or, cette règle n'est en fait qu'une instanciation d'une des règles du Calcul de Lambek, à savoir de la règle générale (voir plus haut) : 'z → z/(y\z)', avec [z :=S] et [y :=N]. Le résultat de cette montée de type au nom propre Luc, conduit à l'opérateur, désigné ici par 'Luc*', ce qui conduit à une nouvelle analyse syntaxique de (b), par la relation (b'') :

(b'') [Luc* :S/(N\S)] + [être-mortel :N\S] → [(Luc*) (être-mortel) :S]

La règle de montée des types traduit simplement que le nom propre 'Luc' peut être analysé soit comme un opérande du prédicat verbal – représentation (b') –, soit comme l'opérateur, 'Luc*', dont le prédicat verbal est devenu l'opérande – représentation (b''). Prenons maintenant la phrase (c) *Tous les hommes sont mortels* ; au syntagme nominal quantifié 'Tous les hommes', qui fonctionne comme un sujet syntaxique, est assigné, conformément à l'analyse logique héritée de Frege, le type syntaxique 'S/(N\S)' d'un opérateur qui s'applique au prédicat verbal 'être mortel' (de type 'N\S') :

(c') [tous les hommes :S/(N\S)]+[être mortel :N\S]
→ [tous les hommes (être-mortel) :S]

Grâce à la règle de montée des types, qui change le type d'une unité linguistique, les sujets syntaxiques (qu'ils soient de simples quantificateurs, des noms propres ou des syntagmes nominaux quantifiés), ont un même comportement syntaxique, exprimé par les analyses catégorielles analogues (a'), (b'') et (c')[26].

Dans le cadre du modèle (AB) des GC, l'analyse syntaxique s'effectue souvent avec de nécessaires retours en arrière (*back tracking*) (voir par exemple l'analyse catégorielle de *Luc admire Marie* présentée précédemment). Dans le Calcul de Lambek, il devient possible d'effectuer une analyse syntaxique incrémentale « de gauche à droite », sans effectuer des retours en arrière. L'analyse syntaxique incrémentale de *Luc admire Marie* s'effectue par un calcul sur les types, en utilisant la règle générale 'z → z/(y\z)' (avec l'instanciation : [z :=S] et [y :=N]), notée [**T**], suivie de la règle générale de composition, notée [**B**], '(z/y)*(y/x) → (z/x)' (avec l'instanciation : [z :=S], [y := S\N] et [x :=N]), et d'une application, pour obtenir finalement le type 'S' :

```
N              +     (N\S)/N        +     N
-----------[T]
S/(N\S)
------------------------------------ [B]
              S/N
-------------------------------------------------------->
                        S
```

De cette analyse syntaxique, on en déduit une organisation applicative des opérateurs linguistiques : le nom propre 'Luc', de type 'N', devient un opérateur 'Luc*', dont le type 'S/(N\S)' se compose avec le type '(N\S)/N' du verbe transitif 'admire', pour former l'opérateur composé 'Luc* **0** admire' de type 'S/N' ; ce dernier, à son tour, s'applique à l'opérande nominal 'Marie' pour former la phrase correcte de type 'S'.

Un certain nombre de travaux – citons par exemple : Moortgat (1988, 1997), Morril (1994), Ranta (1994), Retoré (2001) et les diverses contributions publiés dans Oehrle *et al*. (1988) – ont développé le Calcul de Lambek pour l'appliquer à des analyses syntaxiques de phrases que le modèle AB n'arrivait pas à analyser. Steedman (1988, 2000) a remarqué que la règle [**T**] de montée des types et la règle de composition des types [**B**] du Calcul de Lambek avaient des liens très étroits avec les combinateurs respectifs **C*** et **B** de la Logique Combinatoire. Il en a tiré un certain nombre de consé-

[26] Une autre analyse logique (non frégéenne), de la quantification est en plus grande adéquation avec son expression dans les langues ; elle est présentée dans Desclés (2005 : 289-307) qui introduit les quantificateurs « stars ». Voir aussi Desclés *et al*. (2016b : 283-304).

quences techniques avec ce qu'il a appelé Grammaire Combinatoire (*Combinatory Grammar*). Steedman a introduit également un changement des notations par rapport à celles de Bar-Hillel : soient X un opérateur et XY, ou YX, le résultat de l'application de X à Y selon la position syntagmatique (à droite ou à gauche) de l'opérande Y par rapport à l'opérateur X ; les types et la relation de simplification des types par l'application sont notées :

Ajdukiewicz et Bar-Hillel (AB)	**Steedman**
x/y * y → x	x/y * y → x
y * y\x → x	y * x\y → x

A partir de maintenant, nous utiliserons la notation de Steedman qui a le mérite de laisser invariante la position du type 'x' du résultat dans les notations 'x/y' et 'x\y' du type fonctionnel '**F**yx' et d'indiquer la position attendue du type de l'opérande 'y' par la seule orientation '/' vers la droite ou '\' vers la gauche de la barre d'application.

6. GRAMMAIRE CATÉGORIELLE COMBINATOIRE ET APPLICATIVE (GCCA)

Pour les analyses syntaxiques, nous retiendrons désormais trois types de base : le type N des expressions nominales, le type N* des noms propres et des syntagmes nominaux complets et le type S des expressions applicatives sous-jacentes aux phrase. L'introduction du type N* permet de donner à l'article un rôle particulier de détermination nominale puisqu'il s'applique à un nominal pour constituer un syntagme nominal complet, comme dans l'exemple :

```
une       pomme     rouge
N*/N      N         N\N
        ------------------<
          N
--------------------------->
          N*
```

Le modèle de la Grammaire Catégorielle Combinatoire Applicative (GCCA), développé par I. Biskri et J.-P. Desclés – Biskri (1995), Biskri et Desclés (1996, 1997, 2006) –, exploite systématiquement les liens entre le calcul catégoriel étendu sur les types fonctionnels et la Logique Combinatoire typée (LCT)[27]. En effet, les combinateurs de la LCT étant des opérateurs abstraits de composition et de transformation d'opérateurs quelconques[28], ils se voient assigner des schémas de types fonctionnels qui,

---

27 Les types introduits dans le Logique Combinatoire sont les types fonctionnels ; voir Curry et Feys (1958 : 262-266), Desclés *et al*. (2016a : 201-242).

28 Pour des exemples simples d'utilisation directe des combinateurs en linguistique, voir Desclés *et al*. (2016a : 67-99).

dans un contexte d'opérateurs à composer ou à transformer, se spécifient en types fonctionnels plus particuliers.

Prenons par exemple le combinateur **B** de composition fonctionnelle avec X, opérateur de type **F**βγ, et Y, opérateur de type **F**αβ ; le combinateur **B** construit l'opérateur complexe noté '**B**XY', de type **F**αγ, tel que lorsque le résultat **B**XY s'applique à un opérande 'Z' de type 'α', on obtienne l'expression 'X(YZ)', de type 'γ', après l'élimination de **B**, cette dernière opération indiquant comment ce combinateur **B** agit sur ses opérandes X et Y pour les composer. Dans le style de la déduction naturelle, les règles d'introduction et d'élimination du combinateur **B** sont formulées comme suit :

$$\frac{[X(YZ) : \gamma] \; ; \; [Z : \alpha]}{[\mathbf{B}XY : \underline{\mathbf{F}}\alpha\gamma]} \; [\text{intr. } \mathbf{B}] \qquad \frac{[\mathbf{B}XY : \underline{\mathbf{F}}\alpha\gamma] \; ; \; [Z : \alpha]}{[X(YZ) : \gamma]} \; [\text{elim. } \mathbf{B}]$$

La règle de la montée de type est associée le combinateur **C*** de transformation, dont les règles d'introduction et d'élimination sont :

$$\frac{[XY : \beta] \; ; \; [Y : \alpha]}{[\mathbf{C}^*Y : \underline{\mathbf{F}}(\underline{\mathbf{F}}\alpha\beta)\beta]} \; [\text{intr. } \mathbf{C}^*] \qquad \frac{[\mathbf{C}^*Y : \underline{\mathbf{F}}(\underline{\mathbf{F}}\alpha\beta)\beta] \; ; [X : \underline{\mathbf{F}}\alpha\beta]}{[XY : \beta]} \; [\text{elim. } \mathbf{C}^*]$$

Reprenons l'analyse syntaxique de *Luc admire Marie* dans laquelle le sujet syntaxique 'Luc*' fonctionne comme un opérateur, de type 'S/(S\N*)', obtenu en posant $|\text{Luc}^* =_{\text{def}} \mathbf{C}^*\text{Luc}]$; cet opérateur se compose, au moyen du combinateur **B**, avec l'opérateur 'admire' de type '(S\N*)/N*', d'où l'opérateur composé '**B** Luc* admire' de type 'S/N*'. Avec les spécifications orientées et les notations de Steedman pour les types :

$$[\alpha = N^*] \; ; \; [\gamma = S] \; ; \quad [\underline{\mathbf{F}}\alpha\gamma = S/N^*]$$
$$[\alpha = N^*] \; ; \; [\beta = S] \; ; \quad [\underline{\mathbf{F}}\alpha\beta = S\backslash N^*] \; ; \quad [\underline{\mathbf{F}}(\underline{\mathbf{F}}\alpha\beta)\,\beta = S/(S\backslash N^*)]$$

et en utilisant les règles d'élimination [elim. **B**] et [elim. **C***], nous obtenons l'analyse suivante :

$$\cfrac{[\mathbf{B}\ \text{Luc}^*\ \text{admire} : S/N^*)] \; ; \quad [\text{Marie} : N^*]}{\cfrac{[\text{Luc}^*\ (\text{admire Marie}) : S]}{[\text{Luc}^* : S/(S\backslash N^*) \; ; \quad [\text{admire Marie} : S\backslash N^*]}\ [\text{elim. } \mathbf{C}^*]}\ [\text{elim. } \mathbf{B}]$$
$$[(\text{admire Marie})\ \text{Luc} : S]$$

L'opérateur, noté précédemment 'Luc* **0** admire', est en fait l'opérateur complexe '**B** Luc* admire'. La règle [**T**] de montée d'un type revient à faire appel au combinateur de transformation que Curry a désigné par **C***. Dans l'exemple précédent, les combinateurs **B** et **C*** sont des opérateurs carac-

térisés non seulement par leurs règles d'introduction et d'élimination mais également par des schémas de types fonctionnels[29]. Le combinateur **B** compose deux opérateurs X et Y en construisant un opérateur compos '**B**XY' ; son schéma de type (non orienté) est : '**F**(**F**βγ)**F**(**F**αβ)(**F**αγ)' ; ce schéma indique explicitement que le combinateur **B** doit s'appliquer à un premier opérateur de type '**F**βγ' pour construire un autre opérateur, de type '**F**(**F**αβ)(**F**αγ)' ; cet opérateur qui a pour opérande un opérateur de type **F**αβ construit finalement l'opérateur composé de type **F**αγ . Quant au combinateur **C***, il transforme l'opérande Y, de type α, d'un opérateur X, de type **F**αβ, en un opérateur '**C***Y' de type **F**(**F**αβ)β ; le schéma de type du combinateur **C*** est : '**F**α(**F**(**F**αβ)β'[30].

La GCCA doit non seulement (i) vérifier la bonne correction syntaxique d'une séquence d'unités linguistiques soumise à l'analyse syntaxique, par un calcul effectué uniquement sur les types syntaxiques assignés à ces unités, mais également (ii) construire la représentation applicative (préfixée) d'opérateurs et d'opérandes associée à la séquence syntagmatique analysée. Lorsqu'elle compose des types ou transforme un type par les règles appropriées, l'analyse syntaxique de la séquence syntagmatique (organisée par une concaténation) introduit, dans une première étape, des combinateurs dans l'expression applicative en cours de construction; dans une seconde étape, les combinateurs introduits à l'étape précédente agissent dans l'expression applicative obtenue sur leurs opérandes, ce qui revient à appliquer leurs règles d'élimination ; lorsque tous les combinateurs ont été éliminés après avoir agi sur leurs opérandes, l'expression applicative qui en résulte est ainsi associée à la séquence syntagmatique de départ. Reprenons une fois de plus la séquence Luc admire Marie. En notant par '+' la concaténation des unités linguistiques, nous obtenons la relation entre sa présentation syntagmatique et sa représentation applicative associée :

Luc+admire+Marie → (admire Marie) Luc.

Cette relation se construit dans le cadre de la GCCA par la déduction qui élimine les combinateurs introduits aux différentes étapes de l'analyse syntaxique :

---

29 Sur les schémas de types des combinateurs élémentaires avec des figures associées, voir Desclés *et al*. (2016a : 235-242).

30 Au constructeur '**F**αβ' des types fonctionnels correspondent les types infixées orientées 'β/α' ou 'β\α' ; les types contextualisés de **B** et de **C*** dans l'exemple *Luc admire Marie* sont exprimés (avec les notations de Steedman) par respectivement : '((S/N*) / ((S\N*)/ N*)) / (S/ (S\N*))' et '(S / (S\N*)) / N*'.

1.	[Luc :N*]+[admire :(S\N*) / N*]+[Marie :N*]	*hyp.*
2.	[**C*** Luc :S/ (S\N*)]+[admire :(S\N*)/ N]+[Marie :N*]	*[intr. **C***], 1.*
3.	[**B** (**C***Luc) admire :S/N*]+[Marie :N*]	*[intr. **B**], 2.*
4.	[(**B** (**C***Luc) admire) Marie :S]	*applic. 3.*

5.	(**B** (**C***Luc) admire) Marie	*répétition 4.*
6.	(**C***Luc) (admire Marie)	*[elim. **B**], 5.*
7.	(admire Marie) Luc	*[elim. **C***], 6.*

Commentaire : Les pas 2. et 3. sont obtenus en transformant le type N* du sujet syntaxique puis en composant les types syntaxiques, ce qui revient à introduire le combinateur **C*** puis le combinateur **B**. Le pas 4. est le résultat d'une application de l'opérateur composé à son opérande, d'où l'expression applicative obtenue de type 'S', ce qui démontre que la séquence analysée est une phrase. Il s'agit maintenant d'éliminer à partir du pas 5. les combinateurs dans les expressions applicatives qui sont toutes du type 'S'. Les pas 6. et 7. résultent des règles d'élimination des combinateurs **B** et **C***, d'où l'expression applicative préfixée '(admire Marie) Luc' dans laquelle chaque unité linguistique est soit un opérateur qui s'applique directement à son opérande, soit un opérande absolu. L'expression applicative du pas 7. représente la séquence syntagmatique concaténée 'Luc + admire + Marie' sous la forme d'une expression applicative de même type.

La GCCA étend le lien formel entre le calcul sur les types syntaxiques (composables ou transformables par montée des types) et les deux combinateurs élémentaires **B** et **C***, aux combinateurs élémentaires de la Logique Combinatoire Typée, notamment aux combinateurs[31] :

– de composition :
Φ (combinateur de composition en parallèle),
Ψ (combinateur de distribution),
S (combinateur de fusion),

– et de transformation :
I (combinateur d'identité),
K (combinateur d'abstraction),
W (combinateur de duplication),
C (combinateur de conversion).

Prenons, à titre d'exemple simple, l'analyse syntaxique de *Jean, il admire Marie* avec une thématisation du sujet et l'introduction du clitique 'il'[32] :

---

31 Voir en annexe la liste des combinateurs élémentaires avec leurs schémas de type respectifs ; sur les démonstrations, voir Desclés *et al*. (2016a : 235-242).

32 La présence du clitique 'il' est obligatoire puisque '*Luc, admire Marie' est une séquence mal formée.

```
Luc             il              admire          Marie
N*              α/α             (S\N*)/N*       N*
----------C*    α = (S\N*)
S/(S\N*)
-------------------------------- B
          S/(S\N*)
------------------------------------------------ B
                  S/N*
------------------------------------------------------------------------------->
                                    S
```

Après un changement de type introduit par le combinateur **C***, le sujet thématisé est un opérateur de type 'S/(S\N*)'. Le type '(S\N*)/(S\N*)', assigné au clitique 'il', fonctionne comme un déterminant du prédicat unaire, de type 'S\N*'. La composition du sujet thématisé avec le clitique 'il', introduit le combinateur **B**, d'où l'opérateur complexe '**B** (**C***Luc) il' qui, en se composant avec 'admire', construit le prédicat unaire '**B** (**B** (**C***Luc) il) admire' ; ce prédicat est ensuite appliqué à son opérande 'Marie', pour obtenir l'expression applicative, de type 'S', ayant donc le statut de phrase :

B (**B** (**C***Luc) il) admire) Marie.

L'analyse sémantique du clitique 'il' dans la phrase *Luc, il aime Marie*, est précisée par la déduction suivante obtenue en partant du résultat de l'analyse syntaxique :

1.	**B** (**B** (**C***Luc) il) aime Marie	*hyp.*
2.	**B** (**C***Luc) il) (admire Marie)	*[elim. **B**], 1.*
3.	(**C***Luc) (il (admire Marie))	*[elim. **B**], 2.*
4.	(il (admire Marie)) Luc	*[elim. **C***], 3.*
5.	[il $=_{def}$ **C**(**BBC***) **K** x]	*définition*
6.	((**C**(**BBC***) **K** x) (admire Marie)) Luc	*rempl. 5., 4.*
7.	((**BBC*** x **K**) (admire Marie)) Luc	*[elim. **C**], 6.*
8.	((**B** (**C***x) **K**) (admire Marie)) Luc	*[elim. **B**], 7.*
9.	((**C***x) (**K** (aime Marie)) Luc	*[elim. **B**], 8.*
10.	(**K** (admire Marie) x) Luc	*[elim. **C***], 9.*
11.	(**K** (admire Marie) Luc) Luc	*subst. Luc, 10.*
12.	(admire Marie) Luc	*[elim. **K**], 11.*

Commentaire : Le pas 1. est le résultat de l'analyse syntaxique effectuée dans le cadre de la GCCA. Les pas 2., 3. et 4. sont les résultats des éliminations des occurrences des combinateurs **B** et **C***. Le pas 5. exprime la définition du clitique 'il'. Les pas 6. à 10. sont les résultats des éliminations des combinateurs qui entrent dans la définition de 'il'. Le pas 11. est obtenu après la substitution de l'opérande 'Luc' à la variable 'x' dans le prédicat unaire '**K** (aime Marie) x'. L'élimination du combinateur **K** conduit au pas

12., la forme normale (sans aucune occurrence d'un combinateur) associée à l'expression applicative donnée au pas 1.

La signification du clitique 'il' est formulée (pas 3.) à l'aide du combinateur d'abstraction[33] **K**, lequel construit un opérateur plus abstrait (au pas 10.) à partir de l'expression 'admire Marie', en introduisant une variable 'x' quelconque de type 'N*'. En effet, 'admire Marie' est un prédicat unaire, de type '**F**N*S' ; l'introduction du combinateur **K** construit le prédicat binaire '**K**(admire Marie)', de type '**F**N***F**N*S' ; après avoir été appliqué à la variable 'x' (de référence indéterminée), on obtient le prédicat unaire '**K**(aime Marie) x', de type '**F**N*S', qui, après que 'Luc' ait été substitué à la variable 'x', devient le prédicat unaire '**K**(aime Marie) Luc', ce qui engendre l'expression '(**K**(aime Marie) Luc) Luc', de type 'S' ; l'élimination de **K** construit '(aime Marie) Luc', également de type 'S'. On peut démontrer que le type syntaxique du clitique 'il', dans sa définition [il $=_{def}$ **C**(**BBC***) **K** x] (au pas 5.), est celle d'un déterminant de prédicat unaire, de type '**F**(**F**N*S) (**F**N*S)' assigné par l'analyse syntaxique, à savoir 'α/α' avec [α = S\N*]. Nous obtenons ainsi les relations de réduction :

Luc, il admire Marie
→ **B**(**B**(**C***Luc) il) admire) Marie
→ (**K**(admire Marie) x) Luc
→ (admire Marie) Luc

L'expression '(**K**(admire Marie) x) Luc', de type 'S', fait apparaître le rôle complexe du clitique 'il' dans la thématisation du sujet puisque 'il' est la trace de la variable 'x' dans le prédicat unaire '**K**(admire Marie) x', dérivé par abstraction du prédicat '(admire Marie)', et, en même temps, cette variable 'x' est liée au sujet syntaxique 'Luc' qui peut y être substituée en vue d'obtenir '(admire Marie) Luc'.

Dans le cadre de la GCAC, il est possible d'analyser les différentes formes de thématisations correctes associées à l'expression applicative commune '(admire Marie)'[34] :

Luc, il admire Marie (mais pas : * Luc, admire Marie)
Marie, Luc l' admire (mais pas : * Marie, Luc admire)
Luc, Marie, il l' admire (mais pas : * Luc, Marie admire)
Marie, il l'admire, Luc (mais pas : * Marie, l'admire, Luc) ...

Ces phrases thématisées sont réductibles, par des déductions impliquant des combinateurs, à la même représentation applicative '(admire Marie) Luc' qui constitue la « forme normale » (qui ne possède aucune occurrence

---

[33] Sur le rôle d'abstraction du combinateur **K** et l'analyse de '*il*' dans les constructions impersonnelles, voir Desclés *et al.* (2016a, 91-92).

[34] Il faut, dans ces exemples, tenir compte de la courbe intonative qui est également un marqueur d'une organisation syntaxique et sémantique des énoncés.

d'un combinateur), laquelle est sous-jacente à cette famille d'énoncés apparentés. Le théorème de Church-Rosser[35] affirme que lorsqu'une expression applicative (avec des combinateurs) possède une forme normale sous-jacente, cette dernière est unique, ce qui témoigne de l'intérêt du formalisme de la LCT et de sa robustesse pour son utilisation dans différents domaines[36], dont celui de la linguistique. Avec des développements techniques (linguistiques et formels) plus importants, il est devient possible de procéder à des analyses syntaxiques et sémantiques de problèmes linguistiques plus complexes comme l'analyse syntaxique et sémantique[37] de :

– certains prédicats complexes :
Luc embrasse tendrement Marie / Luc embrasse Marie tendrement.
Luc aime sa (propre) femme, Paul aussi / Luc aime sa femme et Paul aussi (la femme de Luc)
Quand Pedro possède un âne, il le bat.

– les phrases avec coordination :
Luc admire Marie et déteste Sophie.
Jean aime les mathématiques et Pierre la linguistique.
Luc donne à un chien un os et à un policier une rose.
Le drapeau est blanc et rouge.

– les phrases avec relatives déterminatives et explicatives :
Luc qui admire Marie, déteste Sophie.
L'homme qui admire Marie déteste Sophie.

– les phrases avec marqueurs de réflexivisation :
Satan se déteste lui-même.
Satan est le seul à ne détester que lui-même.

– des phrases avec constituants discontinus :
Luc a achet-é une voiture.
Luc ne mange jamais des épinards.
Luc a, malheureusement, achet-é une voiture.

La réduction paraphrastique de ces énoncés à des énoncés sources s'effectue dans le cadre des modèles de la Grammaire Applicative et Cognitive (GAC) et de la GRammaire Applicative des opérations Cognitive et Enonciatives (GRACE)[38] qui ont pour module syntaxique des entrées les résultats des analyses syntaxiques effectuées dans le cadre de la GCCA. Ce genre

35 Voir Desclés *et al*. (2016a : 165-170 ; 2016b : 181-194) qui explicite l'utilisation du théorème de Church-Rosser en linguistique.

36 Par exemple, le Traitement Automatique des Langues (TAL) et l'étude de la sémantique des langages de programmation en informatique théorique.

37 Voir Desclés *et al*. (2016b : 340-418) et les articles publiés dans ce numéro.

38 Voir Desclés (1990) et Desclés *et al*. (2016b).

d'analyses a été appliqué à d'autres langues non indo-européennes (arabe, coréen...)[39].

7. REMARQUES CONCLUSIVES

Des considérations précédentes, nous pouvons tirer un certain nombre de conclusions générales et de constats sur l'activité de langage qui s'exprimé par les langues.

Premier constat. – Le français et un grand nombre d'autres langues sont analysables sous la forme de systèmes sémiotiques composés de différents types d'unités qui sont soit des opérandes absolus, soit des opérateurs appliqués à des opérandes appropriés dans la construction de mots, des syntagmes, des phrases et des énoncés.

Second constat. – Les différentes GC (simples et étendues) analysent les catégories syntaxiques (et également morphologiques) comme étant des types fonctionnels (orientés) de Church : le type fonctionnel d'un opérateur indique à la fois le type d'opérande auquel il peut s'appliquer et le type du résultat de cette application.

Troisième constat. – L'analyse syntaxique n'est pas seulement la simple vérification de la correction syntaxique d'une séquence syntagmatique d'unités linguistiques mais aussi la construction d'une représentation applicative (d'opérateurs appliqués à des opérandes) associée à la séquence syntagmatique qui est reconnue comme étant syntaxiquement une phrase.

Quatrième constat. – Dans le cadre des calculs syntaxiques catégoriels, les types fonctionnels peuvent se composer entre eux, traduisant ainsi une composition de leurs instances linguistiques ; un type d'une unité linguistique peut également se voir être associé à d'autres types plus complexes par des règles de changements de types, traduisant ainsi des changements de fonctionnement syntaxique de cette unité. Dans chacun des cas de composition et de transformation des types, un combinateur de la Logique Combinatoire Typée (LCT) est introduit dans l'expression applicative en cours de construction.

Cinquième constat. – Il y a des interactions théoriques très étroites entre la prise en compte des catégories syntaxiques et morphologiques des langues et les expressions applicatives typées construites et analysées par la Logique Combinatoire Typée (LCT) de Curry (voir par exemple l'isomorphisme de Curry-Howard). Les analyses morpho-syntaxiques puis les analyses sémantiques ultérieures font naturellement appel aux combinateurs de la LCT, comme le prouvent les emplois (parfois seulement implicites) de cette logique dans plusieurs modèles linguistiques qui mettent en œuvre des opérateurs – par exemple : Montague (1974), Shaumyan (1977, 1987), Harris (1976, 1982), Oehrle *et al*. (éds) (1988), Desclés (1990, 2009), Desclés *et al*.

---

39 Voir par exemple Biskri (1995), Kang et Desclés (2008) et les articles de ce numéro.

(2016b). L'utilisation de la programmation applicative (avec des langages de programmation fonctionnelle comme CAML ou HAKELL par exemple) paraît être tout à fait indiquée pour entreprendre des analyses automatiques des langues ancrées sur des conceptualisations théoriques qui éclaireraient le fonctionnement de l'activité de langage et sa spécificité par rapport aux autres systèmes sémiotiques de communication.

Sixième constat. – Les analyses syntaxiques effectuées dans le cadre des GC (simples) et surtout dans le cadre des GC étendues (en particulier le modèle de la GCCA), construisent des expressions applicatives associées aux phrases analysées. Ces analyses permettent ultérieurement d'une part, d'entreprendre des analyses grammaticales des différentes diathèses (constructions actives, passives, réfléchies, impersonnelles...) – voir Desclés, Guentchéva et Shaumyan (1985, 1986) ; Desclés (1990) – et des différentes opérations de prise en charge énonciative (par exemple, aspectuelles et temporelles d'un contenu propositionnel) – voir Desclés et Ro (2011) ; Desclés *et al*. (2016b : 499-531) – et d'autre part, de construire des représentations sémantiques sous la forme d'instances de schèmes sémantico-cognitifs représentés également par des expressions applicatives – voir Desclés (1990), Desclés *et al*. (2016a,b).

Septième constat. – Il est nécessaire de préciser l'articulation entre les différents niveaux d'analyse (morpho-syntaxique, grammaticale, sémantique, sémantico-cognitive, pragmatique) des énoncés au sein d'architectures globales dans lesquelles l'analyse par des Grammaires Catégorielles étendues trouvent place en tant que modules. Les modèles de la GAU, de la GAC et de la GRACE sont des propositions théoriques, encore en développement, allant dans cette direction.

RÉFÉRENCES

AJDUKIEWICZ K. (1935). Die syntaktische Konnexität. *Studia Philosophica* 1, 1-27. Trad. anglaise : Syntactic connexion. In : S. McCall (ed*.), Polish Logic, 1920-1939*. Oxford : Oxford University Press, 1967, 207-231.

BACH E. (1988). Categorial Grammars as theories of language. In : R.T. Oehrle *et al.* (1988), 17-34.

BAR-HILLEL Y. (1950). On syntactic categories. *Journal of Symbolic Logic* 15, 1-16.

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

BENVENISTE E. (1966). « Être » et « avoir » dans leurs fonctions linguistiques. *Problèmes de linguistique générale*, 1, Paris : Gallimard, 187-207.

VAN BENTHEM J. (1988). The Lambek calculus, In : R.T. Oehrle *et al.* (1988), 35-68.

BISKRI I. (1995). *La Grammaire Catégorielle Combinatoire Applicative dans le cadre de la Grammaire Applicative et Cognitive*. Thèse de Doctorat, EHESS, Paris.

BISKRI I., DESCLÉS J.-P. (1996). La Grammaire Catégorielle Combinatoire Applicative. *T.A.L.* 37, n°2, 87-93.

BISKRI I., DESCLÉS J.-P. (1997). Applicative and Combinatory Categorial Grammar (from syntax to functional semantics). In : R. Mitkov, N. Nicolov (eds), *Recent Advances in Natural Language Processing*. Amsterdam : J. Benjamins, 71-84.

BISKRI I., DESCLÉS J.-P. (2006). Coordination, subordination et logique typée. *Faits de langue* 28, 57-66.

BOURDEAU M. (2002). La genèse des grammaires catégorielles et leur arrière-plan logico-philosophique : quelques remarques. In : Godart-Wendling (2002), 13-27.

BUSZKOWSKI W. (1988). Generative power of categorical grammars. In : R.T. Oehrle *et al.* (1988), 69-94.

CASADIO C. (1988). Semantic Categories and the Development of Categorial Grammars. In : R.T. Oehrle *et al.* (1988), 95-123.

CASADIO C. (2002). La logique linéaire non commutative et le calcul de Lambek. In : Godart-Wendling (2002), 93-110.

CHURCH A. (1940). A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic* 5, 56-68.

CHURCH A. (1941). *The Calculi of Lambda-Conversion*. Princeton : Princeton University Press.

CULIOLI A., DESCLÉS J.-P., KABORE R. et KOULOUGHLI D. (1981). *Systèmes de représentations linguistiques et métalinguistiques. Les catégories grammaticales et le problème de langues peu étudiées*. Rapport présenté à l'UNESCO (1980), 141 p., Université de Paris 7, (ERA 642).

CURRY H. (1934). Functionality in Combinatory Logic. *Proceedings of the National Academy of Science of the USA* 20, 584-590.

CURRY H. (1958). Grammatical interpretation of functionality. In : Curry et Feys (1958), 274-275.

CURRY H. (1961). Some logical aspects of grammatical structures. In : R. Jakobson (ed.), *Proceedings of Symposia in Applied Mathematics*, Vol. XII. Providence, Rhode Island : America Mathematical Society, 56-68.

CURRY H., FEYS R. (1958). *Combinatory Logic*, vol 1. North-Holland.

DESCLÉS J.-P. (1990). *Langages applicatifs, langues naturelles et cognition*. Paris : Hermès.

DESCLÉS J.-P. (2003). Analyse Syntaxique et Analyse Discursive. In : D. van Raemdonck (ed.), *Modèles syntaxiques : la syntaxe à l'aube du XXI^e siècle*. Berne : Peter Lang.

DESCLÉS J.-P. (2005). Une analyse non frégéenne de la quantification. In : P. Joray (éd.), *La quantification dans la logique moderne*. Paris : L'Harmattan, 263-312.

DESCLÉS J.-P. (2009) Le concept d'opérateur en linguistique. *Histoire, épistémologie, langage* XXXI/1, 75-98.

DESCLÉS J.-P. (2016). Les mathématiques de la grammaire d'opérateurs de Zellig Harris. In : C. Martinot *et al.* (éds), *Perspectives harrissiennes*. Paris : Cellule de Recherche en Linguistique, 83-105.

DESCLÉS J.-P., GUENTCHÉVA Z. et SHAUMYAN S. (1985). *Passivization in Aplicative Grammar*. Amsterdam : J. Benjamins.

DESCLÉS J.-P., GUENTCHÉVA Z. et SHAUMYAN S. (1986). A theoretical analysis of reflexivisation in the framework of Applicative Grammar. *Linguisticae Inverstigationnes* 10/1, 1-65.

DESCLÉS J.-P., SEGOND F. (1992). Topicalization : Categorial Analysis and Applicative Grammar. In : Lecomte (1992), 13-37.

DESCLÉS J.-P., BISKRI I. (1995). Logique combinatoire et linguistique : la Grammaire Catégorielle Combinatoire Applicative. *Mathématiques, Informatiques et Sciences Humaines* 132, 39-68.

DESCLÉS J.-P., DJIOUA B. et LE PRIOL F. (2010). *Logique et langage : déduction naturelle.* Paris : Hermann.

DESCLÉS J.-P., RO H. (2011). Opérateurs aspecto-temporels et logique combinatoire. *Mathématiques et Sciences Humaines* 194, 39-70.

DESCLÉS J.-P., GUIBERT G. et SAUZAY B. (2016a). *Logique combinatoire et λ-calcul : des logiques d'opérateurs*. Toulouse : Cépaduès.

DESCLÉS J.-P., GUIBERT G. et SAUZAY B. (2016b). *Calculs de significations par une logique d'opérateurs.* Toulouse : Cépaduès.

EILENBERG S., MAC LANE S, (1945). General Theory of Natural Equivalences. *Transactions of the American Mathematical Society* 58, 231-294.

FREGE G. (1891). *Funktion, Begriff, Bedeutung*. Göttingen : Vandenhoeck & Ruprecht.

FREGE G. ([1893], 1967). *Grundgesetze der Arithmetik, vol. 1*. Jena : H. Pohle. Trad. anglaise : *Basic Laws of Arithmetic, Exposition of the systems*. Berkeley, Los Angeles : University of California Press.

FREGE G. (1971). *Écrits logiques et philosophiques*. Paris : Seuil.

GARDIES J.-L. (1975). *Esquisse d'une grammaire pure*. Paris : Vrin.

GINISTI J.-P. (1997). *La logique combinatoire*. Paris : Presses Universitaires de France.

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

GODART-WENDLING B. (éd.) (2002). Les grammaires catégorielles. *Langages* 148.

GODART-WENDLING B. (2002). Les trois premières grammaires catégorielles. *Langages* 148, 51-66.

GROSS M., LENTIN A., (1967). *Notions sur les grammaires formelles*. Avec une préface de Noam Chomsky. Paris : Gauthier-Villars.

GUENTCHÉVA Z. (1976). *Présentation critique du modèle applicationnel de S.K. Shaumyan*. Paris : Dunod.

HARRIS Z. (1951). *Methods in Structural Linguistic*. Chicago : Chicago Press.

HARRIS Z. (1976). *Notes du cours de syntaxe*. Paris : Seuil.

HARRIS Z. (1982). *A Grammar of English. Based on Mathematical Principles*. New York : John Wiley.

VAN HEIJENOORT (ed.) (1971). *From Frege to Gödel. A Source Book in Mathematical Logic, 1879-1931*. Cambridge, MA : Harvard University Press.

HINDLEY J.R., SELDIN J.-P. (2008). *Lambda-calculus and combinators : an introduction*. New York : Cambridge University Press.

HUSSERL E. ([1913], 1972). *Logische Untersuchungen, III, IV*. Halle : Max Niemeyer Verlag. Trad. française : *Recherches logiques*. Paris : Presses Universitaires de France.

JESPERSEN O. (1971). *La philosophie de la grammaire*. Paris : Éditions de Minuit.

JORAY P., GODART-WENDLING B. (2002). De la catégorie sémantique de Leśniewski à l'analyse de la quantification dans la syntaxe d'Ajdukiewicz. *Langages* 148, 28-50.

KANG J.Y., DESCLÉS J.-P. (2008). Korean Parsing Based on the Applicative Combinatory Categorial Grammar. *The 22nd PACLIC*, Cebu city, Phillippines.

LAMBEK J. (1958). The Mathematics of Sentence Structure. *American Mathematical Monthly* 65, 154-170.

LAMBEK J. (1961) On the calculus syntactic types. In : R. Jakobson (ed.), *Proceeding of symposia in Applied Mathematics, vol. XII*. Providence, Rhode Island : American Mathematical Society, 166-178.

LAMBEK J. (1980). From Lambda calculus to Cartesian closed categories. In : R. Hindley, J. Seldin (eds), *To H.B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism*. London : Academic Press, 375-402.

LAMBEK J. (1988). Categorial and Categorical Grammars, In : Oehrle *et al.* (1988), 297-317.

LAWVERE F.W., SCHANUEL S.H. (2009). *Conceptual Mathematics, A first introduction to categories*. Cambridge University Press.

LECOMTE A. (ed.) (1992). Word Order in Categorial Grammar. *Proceedings of the Workshop of Clermont-Ferrand on behalf of DYANA project*, May 25-27, 1990. Clermont-Ferrand : ADOSA.

LEŚNIEWSKI S. (1929). Grundzüge eines neues System der Grundlagen der Mathematik. *Fundamenta Mathematicae* 14/1, 1-81.

MAC LANE S. (1971). *Categories for the Working Mathematician*. New York : Springer Verlag.

MONTAGUE R. (1970). Universal Grammar. *Theoria* 36, 373-398. Reprinted in Montague (1974), 222-246.

MONTAGUE R. (1973). The proper treatment of quantification in ordinary English. In : K.J.J. Hintikka, J.M.E. Moravcsik, P. Suppes (eds), *Approaches to Natural Language*. Dordrecht : Reidel, 221-242. Reprinted in Montague (1974), 247-270.

MONTAGUE R. (1974). *Formal philosophy : selected papers of Richard Montague*, R.H. Thomason (ed.). New Haven : Yale University Press.

MOORTGAT M. (1988). *Categorial Investigation. Logical and Linguistic Aspects of the Lambek calculus*. Dordrecht : Foris Publications.

MOORTGAT M. (1997). Categorial Type Logics. In : J. Van Benthem et A. ter Meulen (eds), *Handbook of Logic and Language*, Chapter 2. Elsevier.

MORRILL G. (1994). *Type-Logical Grammar*. Dordrecht : Kluwer.

OEHRLE R.T. BACH E., et WHEELER D. (1988). *Categorial Grammars and Natural Languages Structures*. Dordrecht : Reidel.

RANTA A. (1994). *Types-Theoretical Grammar*. New York : Oxford University Press.

RETORÉ C. (2001). Systèmes déductifs et traitement des langues : un panorama des grammaires catégorielles. *Technique et science informatiques* 20, Paris, 301-336.

RUSSELL B. (1908). Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics* 30/3, 222-262. Reprinted in : Van Heijenoort (1971), 150-182.

SCHÖNFINKEL M. ([1924], 1971). On the Building Blocks of Mathematical Logic. In : Van Heijenoort (1971), 355-366.

SHAUMYAN S. (1965). *Strukturnaja linguistika*. Moscow : Nauka.

SHAUMYAN S. (1977). *Applicational Grammar as a Semantic Theory of Natural Language.* Chicago : University of Chicago Press.

SHAUMYAN S. (1987). *A Semiotic Theory of Natural Language*. Bloomington : Indiana University Press.

STEEDMAN M. (1988). Combinators and Grammars. In : Oehrle *et al.* (1988), 207-263.

STEEDMAN M. (1989). *Work in progress : Combinators and grammars in natural language understanding*. Summer institute of linguistic, Tucson University.

STEEDMAN M. (2000). *The Syntactic Process*. Cambridge : MIT Press / Bradford Books.

TESNIÈRE L. ([1959],1988). *Éléments de syntaxe structurale*. Paris : Klincksieck.

ANNEXE

Combinateurs élémentaires

Combinateurs de composition		Introduction	Élimination	Schéma de type
B	de composition fonctionnelle	X(Yu) ---------- (**B**XY) u	(**B**XY) u ---------- X(Yu)	**F**(**F**βγ) (**F**(**F**αβ)(**F**αγ))
S	de fusion	Xu(Yu) ---------- (**S**XY) u	(**S**XY) u ---------- Xu(Yu)	**F**(**F**α(**F**βγ)) (**F**(**F**αβ)(**F**αγ))
Φ	de composition en parallèle	X(Yu)(Zu) ---------- (**Φ**XYZ) u	(**Φ**XYZ) u ---------- X(Yu)(Zu)	**F**(**F**β**F**γδ))(**F**(**F**αβ)**F**(**F**αγ)(**F**αδ))
Ψ	de distribution	X(Yu)(Yv) ---------- (**Ψ**XY) uv	(**Ψ**XY) uv ---------- X(Yu)(Yv)	**F**(**F**β**F**βγ))(**F**(**F**αβ)**F**(**F**α**F**αγ)))

Combinateurs de transformation

I	d'identité	X ----- **I**X	**I**X ----- X	**F**αα
K	d'abstraction	X --------- (**K**X) u	(**K**X) u --------- X	**F**β(**F**αβ)
W	de diagonalisation	Xuu --------- (**W**X) u	(**W**X) u ---------- X uu	**F**(**F**α**F**αβ)) (**F**αβ)
C	de conversion	Xuv ---------- (**C**X) vu	(**C**X) vu ---------- Xuv	**F**(**F**α(**F**βγ)) (**F**β(**F**αγ))
C*	de montée	XY ---------- (**C***Y)X	(**C***Y) X ---------- XY	**F**α(**F**(**F**αβ)β)