MATHÉMATIQUES ET SCIENCES HUMAINES

JEAN-PIERRE DESCLÉS
ISMAIL BISKRI

Logique combinatoire et linguistique : grammaire catégorielle combinatoire applicative

Mathématiques et sciences humaines, tome 132 (1995), p. 39-68
<http://www.numdam.org/item?id=MSH_1995__132__39_0>

Math., Inf. Sci. hum., (33e année, n° 132, *1996*, p. 39-68)

LOGIQUE COMBINATOIRE ET LINGUISTIQUE : GRAMMAIRE CATÉGORIELLE COMBINATOIRE APPLICATIVE

Jean-Pierre DESCLÉS[1], Ismaïl BISKRI[1]

RÉSUMÉ — *La Grammaire Catégorielle Combinatoire Applicative étend la Grammaire Catégorielle Combinatoire de Steedman par une association canonique entre les règles et des combinateurs de Curry d'une part et l'utilisation de métarègles qui contrôlent les opérations de changement de type d'autre part. Ce modèle est inclus dans le modèle général de la Grammaire Applicative et Cognitive (Desclés) avec trois niveaux de représentation : (i) le phénotype (expressions concaténées) ; (ii) le génotype (expressions applicatives) ; (iii) les représentations cognitives (sens des prédicats linguistiques). L'objectif de cet article est : (i) l'analyse automatique des expressions phénotypiques ; (ii) la construction des expressions applicatives sous jacentes aux énoncés analysés. L'analyse théorique est appliquée aux problèmes de la pseudo-ambiguité et de la coordination.*

SUMMARY — Combinatory logic and linguistics : applicative and combinatory categorial grammar
Applicative and Combinatory Categorial Grammar is an extension of Steedman's Combinatory Categorial Grammar by a canonical association between rules and Curry's combinators on one hand and metarules which control type-raising operations on the other hand. This model is included in the general framework of Applicative and Cognitive Grammar (Desclés) with three levels of representation : (i) phenotype (concatened expressions) ; (ii) genotype (applicative expressions) ; (iii) the cognitive representations (meaning of linguistic predicates). The aim of the paper is : (i) an automatic parsing of phenotype expressions that are underlying to sentences ; (ii) the building of applicative expressions. The theoretical analysis is applied to spurious ambiguity and coordination.

1. LINGUISTIQUE ET THÉORIE DE LA DÉMONSTRATION

Sous sa *forme observable* (représentation concaténée), un énoncé peut être considéré comme une suite de mots ordonnés selon des règles syntagmatiques qui ne reflètent pas forcément l'ordre dans lequel les mots s'appliquent les uns aux autres pour construire l'interprétation sémantique fonctionnelle. Cette dernière est exprimée par une structure sous-jacente à l'énoncé (pour nous : une structure applicative), construite au moyen de l'*opération fondamentale de l'application d'un opérateur à un opérande*. Il n'est pas toujours facile de reconstituer la structure applicative sous-jacente à un énoncé observable, car plusieurs phénomènes de structurations linguistiques sont intriqués dans la chaîne syntagmatique.

[1] Centre d'Analyse et de Mathématique Sociales, équipe de recherche LaLIC, Université de Paris-Sorbonne.

Nous reprenons une distinction proposée par S. K. Shaumyan (1965, 1977, 1987), reprise puis étendue ultérieurement par J. P. Desclés (1990). Une telle distinction[1] met en parallèle deux niveaux de représentation et d'analyse des langues : le phénotype et le génotype.

- Le *niveau du phénotype* est ce que nous pourrions appeler un "niveau de surface" (représentation concaténée) assez directement observable. Il est caractérisé par l'ensemble des traits spécifiques à une langue particulière (morphologie, syntaxe, *etc.*). Il fait apparaître entre autres en particulier l'ordre et la position des mots dans une phrase.

- Le *niveau du génotype* , selon S.K. Shaumyan, comprend une partie organisée comme un langage formel appelé "le langage génotype". S.K. Shaumyan cherche à décrire et à formuler tous les *invariants sémiotiques* constitutifs des langues naturelles sous forme d'opérations et de relations. En outre, l'interprétation sémantique fonctionnelle d'une phrase est représentée par une expression applicative du langage génotype. Le langage génotype est alors engendré par une grammaire dite "applicative" qui se veut "universelle". Ce dernier terme est pris au double sens suivant : (i) caractériser des invariants et établir des *formulations universelles* des catégories grammaticales des langues (c'est à dire indépendantes des encodages morpho-syntaxiques dans une langue particulière) ; (ii) caractériser de manière idéale la fonction langagière qui "s'incarne" dans chaque langue naturelle spécifique en établissant un morphisme du langage génotype vers les différentes langues phénotypiques. Le langage génotype est encodé dans chaque langue particulière par des procédures spécifiques. Chaque langue est représentée au niveau du génotype par des expressions métalinguistiques. Nous sommes amenés à adopter la démarche (Desclés, 1990 : 216) qui associe à toute phrase d'une langue naturelle une expression applicative : *étant donnée une langue naturelle (LN) que l'on peut considérer comme un ensemble de phrases empiriques (au moins en première approximation), chaque phrase de LN est alors représentée par une expression applicative d'un système applicatif.*

Nous ne considérons pas les énoncés comme de simples suites concaténées de mots, mais comme des organisations d'unités linguistiques qui fonctionnent comme des opérateurs appliqués à des opérandes. Cette organisation applicative n'est pas directement observable dans le phénotype sous-jacente aux phrases et aux énoncés ; elle devient cependant explicite dans le génotype. En effet, dans le phénotype, les opérateurs ne sont pas toujours contigus à leurs opérandes puisque l'organisation phénotypique des phrases dépend des propriétés d'encodage du génotype dans le phénotype (ordre des mots, morphologie, ...). Cependant, même si l'organisation applicative n'apparaît pas directement au niveau du phénotype, la dichotomie opératoire qui analyse certaines unités linguistiques dans un énoncé comme des opérateurs et d'autres comme des opérandes serait, elle, constitutive du langage. Cette dichotomie devrait nous permettre de reconstituer par une procédure explicite l'ordre applicatif, sous-jacent aux énoncés. Le principe d'une correspondance entre un *ordre concaténé* du phénotype et un *ordre applicatif* du génotype peut être formulé comme suit : tout énoncé est représentable par une expression applicative (opérateur-opérande) qui reconstruit l'ordre applicatif des unités linguistiques concaténées dans l'énoncé.

Les problèmes que nous devons résoudre sont :
1) Quelle est la procédure effective (éventuellement programmables) qui permettrait d'associer à un énoncé sa représentation applicative sous-jacente ?

---

[1] Zlatka Guentcheva-Desclés (1976) associe la distinction phénotype/génotype, proposée par Shaumyan, à la distinction qui existe entre langue et langage. La langue est organisée dans un système linguistique composé essentiellement de signes. Cette organisation est sujette à des critères propres à une communauté linguistique donnée. Le langage est la capacité de l'homme à exprimer et à communiquer des idées. Cette capacité est mise en oeuvre au moyen d'un système de signes vocaux (paroles) ou graphiques.

ii) Étant donnée une expression bien formée du langage génotype, comment engendrer l'énoncé dans une langue phénotypique particulière ?

Le premier problème est une analyse qui conduit à une représentation, le second est un problème de génération. Nous allons, dans cet article, montrer quelles sont les *propriétés formelles* mises en oeuvre dans la procédure effective que nous proposons. Dans la résolution du premier problème, nous souhaitons résoudre en même temps deux sous-problèmes :

i) construire pour chaque énoncé une analyse syntaxique correcte (dans le cadre que nous allons préciser) ;
ii) construire une représentation applicative qui donnera une interprétation fonctionnelle de l'énoncé.

L'interprétation fonctionnelle des deux énoncés élémentaires suivants :

(1) *Jean aime Marie*
(2) *Marie aime Jean*

conduit à deux représentations distinctes. En effet, la langue française place en général le sujet avant le verbe ; cette indication nous permet de comprendre que la première phrase signifie "Jean aime une personne prénommée Marie". La seconde phrase signifie que "Marie aime une personne prénommée Jean". Dans le génotype, nous avons deux constructions applicatives distinctes qui sont deux interprétations sémantiques fonctionnelles de (1) et de (2).

(1') *((aime Marie) Jean)*
(2') *((aime Jean) Marie)*

Dans cette interprétation sémantique fonctionnelle, le "verbe" transitif *aime*, est considéré comme opérateur "à deux places" ; il s'applique à "l'objet" de façon à construire un autre opérateur qui lui même s'applique au "sujet" pour construire la proposition sous-jacente à la phrase.

Un certain nombre de modèles fondés sur les types syntaxiques existent. Le modèle qui correspond à la conceptualisation des unités linguistiques comme opérateur ou opérande, celui des Grammaires Catégorielles, utilise des méthodes logiques pour rendre compte de la syntaxe des langues (et, particulièrement, de la sémantique). Les travaux de Ajdukiewicz (1935), Bar-Hillel (1953), Curry (1958), Lambek (1958, 1961), Montague (1974), Geach (1972), Shaumyan (1987), Steedman (1982, 1989), Moortgat (1988), Bach (1984), utilisent une analyse syntaxique sans règles de réécriture comme les modèles chomskyens ou modèles apparentés le font. Les Grammaires Catégorielles substituent un calcul sur des types à un traitement sur des unités linguistiques. Un intérêt particulier fut réservé aux développements des Grammaires Catégorielles depuis les travaux de M. Steedman (1982, 1989) et d'E. Bach (1981, 1984).

Les Grammaires Catégorielles conceptualisent les langues comme des systèmes d'agencement d'unités linguistiques (mots, morphèmes, lexies, ...) dont certaines fonctionnent comme des opérateurs alors que d'autres fonctionnent comme des opérandes, ce qui se traduit par une assignation à chaque unité (en général lexicale) d'un ou de plusieurs types qui précisent d'une part, si l'unité fonctionne comme un opérateur ou un opérande et d'autre part, comment un opérateur s'applique un opérande.

Les Grammaires Catégorielles présentent un défaut majeur : la pseudo-ambiguïté (*spurious ambiguity*). Une phrase simple non ambiguë peut avoir plusieurs analyses syntaxiques possibles qui ne correspondent qu'à une seule interprétation sémantique fonctionnelle. Prenons l'exemple de la phrase *Jean aime Marie* ; nous pouvons vérifier que le verbe (*aime*) est précédé par un sujet (*Jean*), puis que le constituant formé par ces deux unités (*Jean* et *aime*) est suivi par un objet (*Marie*). Nous pouvons également vérifier que le verbe (*aime*) est suivi par un objet (*Marie*), puis que le constituant formé par les deux unités (*aime* et *Marie*) est précédé par le sujet (*Jean*). Nous avons donc deux calculs possibles qui vérifient que la séquence *Jean aime Marie* est syntaxiquement correcte. Ces deux analyses syntaxiques ont cependant une seule interprétation sémantique, la phrase n'étant pas ambiguë (ni syntaxiquement, ni sémantiquement). Comme le signale (Lecomte, 1994), les différentes analyses issues du problème de la pseudo-ambiguïté correspondent en fait à des déductions différentes dans le système catégoriel employé, mais correspondent à un même "théorème". Il note aussi que ce problème n'est pas propre à la linguistique, car on le retrouve aussi dans la théorie de la démonstration en logique.

Plusieurs voies sont explorées pour résoudre cette question, nous citerons entre autres celle des réseaux de preuve (Roorda, 1992) (Lecomte, 1992, 1993) ou encore celle que nous envisageons : *une stratégie d'analyse quasi-incrémentale de gauche à droite*. Cette dernière stratégie est motivée par plusieurs facteurs[2] :

i) La possibilité de construire une seule déduction parmi l'ensemble des déductions possibles sans nous faire perdre d'informations : toutes les déductions démontrent un même "théorème" correspondant au type d'une phrase syntaxiquement correcte ; cette stratégie élimine la pseudo-ambiguïté.
ii) L'efficacité "computationnelle" . L'efficacité de la procédure est théorique puisque nous construisons une seule déduction ; l'efficacité est pratique puisque la stratégie retenue conduit à une implémentation opératoire économique.

Il a été proposé dans (Biskri, 1995) l'automatisation d'une méthode effective qui relie un énoncé du niveau phénotypique à une analyse applicative sous-jacente exprimée dans le génotype. La Grammaire Catégorielle Combinatoire Applicative (GCCA) est un système[3] que nous allons décrire dans l'article,

(i) reconnaît les phrases syntaxiquement correctes ;
(ii) construit une interprétation sémantique fonctionnelle de l'énoncé. La construction de cette interprétation se fera au moyen des combinateurs de la logique combinatoire de Curry introduits progressivement par des règles d'introduction.

Techniquement, une association canonique sera établie entre des règles inférentielles sur les types (interprétables par des catégories syntaxiques et des combinateurs de la logique combinatoire de Curry (1958).

Une analyse fondée sur la Grammaire Catégorielle Combinatoire Applicative (GCCA) s'effectue en deux grandes étapes :

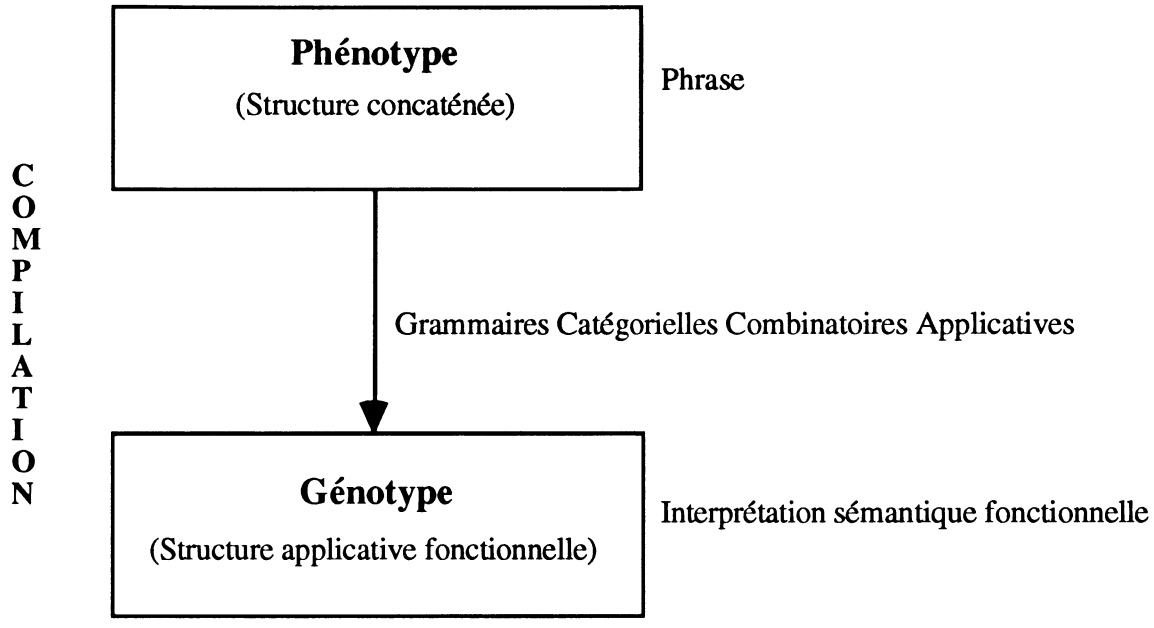
---

[2] Des études obtenues en psychologie semblent montrer que notre compréhension d'une phrase est plutôt quasi-incrémentale, autrement dit l'acquisition du sens est graduelle et chaque mot dans une phrase contribue à cette construction graduelle du sens (Haddock, 1987) (Steedman, 1989) (au moins lorsque nous procédons à une analyse syntaxique de la phrase).

[3] Ce système fonctionnera comme une compilation.

(i) La première étape s'illustre par la vérification de la bonne connexion syntaxique et la construction de structures applicatives avec des combinateurs introduits *progressivement* à certaines positions de la chaîne syntagmatique. Des métarègles seront utilisées pour contrôler le déclenchement de certaines règles d'introduction. L'expression obtenue appartient au langage génotype.

(ii) La deuxième étape consiste à utiliser les règles de β-réduction de la logique combinatoire de façon à construire une autre structure applicative en éliminant successivement tous les combinateurs introduits à la première étape de façon à construire la "forme normale" (c'est à dire irréductible) ; cette dernière exprime l'interprétation sémantique fonctionnelle de l'énoncé d'entrée. Ce dernier calcul s'effectue entièrement dans le génotype.



La GCCA et les combinateurs inscrivent ce travail dans une approche fondée sur une théorie de la preuve considérée comme un outil de traitement linguistique. Nous avons pour objectif l'établissement de liens solides entre linguistique théorique, théorie des catégories, théorie de la démonstration, informatique théorique, afin de faire émerger des propriétés mathématiques et logiques qui seraient sous-jacentes aux structures linguistiques.

2. LES GRAMMAIRES CATÉGORIELLES

Nous retrouvons les origines philosophiques des Grammaires catégorielles dans les travaux du philosophe Husserl (1913). Ce dernier, reprend une opposition traditionnelle depuis les Grecs et distingue les expressions *catégorématiques* des expressions *syncatégorématiques*. Il est en effet impossible d'admettre selon lui que n'importe quel symbole grammatical soit doué d'une signification indépendante des éléments sur lesquels il porte : les syncatégorèmes ont une signification "incomplète". C'est ainsi que les expressions qui sont pourvues d'une signification sont les expressions catégorématiques, elles apparaissent sous formes de syntagmes nominaux ou d'énoncés et les expressions qui ne sont pas pourvues d'un sens complet sont les expressions syncatégorématiques. Avec ces expressions une signification complète n'est obtenue que conjointement avec les significations partielles d'autres parties du discours.

Pour sa part, le logicien polonais Lesniewski a expliqué que sa conception des catégories sémantiques esquissée par lui en 1922 reprenait la tradition des catégories aristotéliciennes, des parties du discours de la grammaire traditionnelle, et enfin, des catégories de signification développée par Husserl. Il a retenu deux sortes d'expressions: les noms et les propositions. Les noms sont des expressions linguistiques qui servent à désigner des entités objectales ou des classes de telles entités. Les propositions sont des expressions linguistiques qui sont construites par une énonciation visant à représenter un "état de chose" comme "objet intentionnel". En dehors de ces deux sortes de catégories fondamentales, les autres expressions du langage seront des syncatégorèmes.

Partant de cette classification, nous pouvons constater que les expressions syncatégorématiques *fonctionnent comme des opérateurs* qui contribuent à constituer des expressions catégorématiques. C'est en reprenant cette classification, ébauchée donc par Husserl et Lesniewski, que K. Ajdukiewicz (1935), puis Y. Bar-Hillel (1953) et enfin J. Lambek (1958, 1961) ont proposé différents systèmes formels pour vérifier la bonne connexion syntaxique des expressions linguistiques. Ces systèmes formels, que nous dirons classiques, sont désignés sous le nom de : "Grammaires Catégorielles". Ils comportent un ensemble de catégories divisées en deux classes : les catégories de base et les différentes catégories d'opérateurs, également appelées foncteurs.

Les Grammaires Catégorielles assignent des catégories aux unités (lexicales et morphèmes) du dictionnaire. La bonne connexion syntaxique est vérifiée en simplifiant successivement les catégories. Une phrase est une unité linguistique qui a une bonne connexion syntaxique qui se simplifie en une catégorie de phrase.

Une Grammaire Catégorielle est donc définie pour la donnée d'une assignation des catégories à ses unités linguistiques. La simplification des catégories revient à effectuer un calcul inférentiel sur les catégories et non pas sur les unités linguistiques. On conceptualise donc les Grammaires Catégorielles comme des *systèmes inférentiels de types* en considérant *chaque catégorie comme un type.*

Une Grammaire Catégorielle est donc déterminée par différents types d'unités linguistiques (on dira différentes unités typées). Toute l'information pour vérifier la bonne connexion des phrases (plus généralement des syntagmes) est donnée par les types assignés aux unités linguistiques de la grammaire.

Système inférentiel de types

On considère donc maintenant les types de base et les types dérivés.

L'ensemble des types est défini comme suit :
(T_0) les types de base sont des types ;
(T_1) Si X et Y sont des types alors X/Y et X\Y sont des types ;
(T_2) Si X et Y sont des types, X-Y est un type.

Une entité "e" (unité linguistique par exemple) qui est de type X, sera notée "[X : e]".

Les types dérivés X/Y et X\Y se lisent respectivement "X sur Y" et "X sous Y". L'interprétation de ces types dérivés est *fonctionnelle* : ils représentent le type d'un opérateur qui a pour argument une entité de type X et qui donne pour résultat une entité de type X. L'interprétation fonctionnelle se déduit des deux *règles applicatives* (avant et arrière) suivantes :

$$(\mathbf{A>})\quad \frac{[X/Y : f]\,,\,[Y : a]}{[X : f - a]}\qquad ;\qquad (\mathbf{A<})\quad \frac{[Y : a]\,,\,[X\backslash Y : f]}{[X : a - f]}$$

Nous nous donnons deux règles d'inférences sur les types, directement associées aux règles applicatives précédentes :

$$(\mathbf{AB>})\quad \frac{X/Y - Y}{X}\qquad ;\qquad (\mathbf{AB<})\quad \frac{Y - Y\backslash X}{X}$$

Les règles signifient que des deux types contigus et présents dans cet ordre dans la prémisse, on en infère le type de la conclusion.

La concaténation, notée '-', des types peut s'interpréter comme une concaténation ou un produit cartésien.

A titre d'exemple élémentaire, assignons aux unités lexicales suivantes leurs catégories :

Jean : N
admire : (S\N)/N
Marie : N

où N désigne la catégorie des unités nominales et S la catégorie des phrases. La vérification de la bonne connexion syntaxique de la phrase *Jean admire Marie* s'effectue comme suit :

Jean	admire	Marie
N	- (S\N)/	- N
N	- S\N	
S		

puis l'on a les simplifications suivantes :

$$N - ((S\backslash N)/N - N) \quad ==> \quad N - S\backslash N \quad ==> \quad S$$

Un *calcul sur les types* est caractérisé par le système suivant :

$$< To\ ,\ -\ ,\ /\ ,\ \backslash\ ,\ \rightarrow\ >^{4}$$

où To est un ensemble de types de base ; -, /, \ sont des opérations constructeurs de types ; $\rightarrow$ est une relation de préordre sur les types de T engendré par les constructeurs à partir de To. La relation $\rightarrow$ est une réduction entre types.

Le *calcul de Lambek* L est un calcul sur les types tel que l'on ait les trois axiomes et les règles d'inférence suivantes :

---

[4] En substituant à '/' ou à '\' le symbol de l'implication '$\supset$' et sans s'occuper de l'ordre des hypothèses de la prémisse, on obtient le schéma du *modus ponens* propositions :

$$\frac{Y \supset X\,,\,Y}{X} \quad \text{ou} \quad \frac{Y\,,\,Y \supset X}{X}$$

Cette analogie est à la base de l'isomorphisme de Howard-Curry.

(L1)	$X \rightarrow X$	Réflexivité
(L2)	$(X\text{-}Y)\text{-}Z \rightarrow X\text{-}(Y\text{-}Z)$	Associativité de la concaténation
(L3)	$X\text{-}(Y\text{-}Z) \rightarrow (X\text{-}Y)\text{-}Z$	Associativité de la concaténation
	$X \rightarrow Y$ et $Y \rightarrow Z$	
(RL1)	----------	Transitivité
	$X \rightarrow Z$	
	$X\text{-}Y \rightarrow Z$	
(RL2)	----------	Curryfication à droite
	$X \rightarrow Z/Y$	
	$X\text{-}Y \rightarrow Z$	
(RL3)	----------	Curryfication à gauche
	$Y \rightarrow Z\backslash X$	
	$X \rightarrow Z/Y$	
(RL4)	----------	Décurryfication à droite
	$X\text{-}Y \rightarrow Z$	
(RL5)	$Y \rightarrow Z\backslash X$	
	----------	Décurryfication à gauche
	$X\text{-}Y \rightarrow Z$	

On en déduit les théorèmes suivants :

(TL1)	$X \rightarrow (X\text{-}Y)/Y$	
(TL2)	$(Z/Y)\text{-}Y \rightarrow Z$	Application
(TL3)	$Y \rightarrow Z\backslash(Z/Y)$	Changement de type
(TL4)	$(Z/Y)\text{-}(Y/X) \rightarrow Z/X$	Composition fonctionnelle de types
(TL5)	$Z/Y \rightarrow (Z/X)/(Y/X)$	Division des types
(TL6)	$(Y\backslash X)/Z \leftrightarrow (Y/Z)\backslash X$	Permutation[5]
(TL7)	$(X/Y)/Z \leftrightarrow X/(Z\text{-}Y)$	
	$X \rightarrow X'$, $Y \rightarrow Y'$	
(TL8)	----------	
	$X\text{-}Y \rightarrow X'\text{-}Y'$	Règles dérivées
	$X \rightarrow X'$, $Y \rightarrow Y'$	
(TL9)	----------	
	$X/Y' \rightarrow X'/Y$	

Ces théorèmes expriment de nouvelles réductions sur les types spécifiques au calcul de Lambek. Tous ces théorèmes sont déduits des axiomes et des règles précédents.

REMARQUE. Outre la réduction applicative qui est exprimée par le théorème (TL2), l'originalité principale du calcul de Lambek se trouve dans l'introduction des théorèmes (TL3) et (TL4). Le théorème (TL3) (changement de type) est appelé dans la littérature "type-raising". Il a été proposé par Lambek dans son analyse des pronoms (Desclès, 1990)[6].

---

[5] M. Hepple (1991) et M. Moortgat (1988) désignent cette règle par : règle d'associativité. L'appelation est dangereuse, l'interprétation fonctionnelle des types n'est pas associative.
Nous verrons dans la section 5 que cette règle introduit plutôt la notion de permutation.

[6] Nous invitons le lecteur à consulter le livre de Jean Pierre Desclès (1990) pour avoir une appréciation plus complète sur le sujet. Il y trouvera certainement plus de détails sur les motivations syntaxiques et sémantiques du changement de type et de la composition.

La Grammaire Catégorielle Combinatoire (Steedman 1982, 1987, 1989) tend à généraliser les Grammaires Catégorielles de Ajdukiewicz, Bar-Hillel et de Lambek. Cette généralisation reprend l'idée des Grammaires Catégorielles classiques, que les unités des langues naturelles sont des foncteurs et des arguments de différents types ; elle se donne en outre certaines opérations sur les types (opérations unaires et binaires).

Les opérations[7] que nous décrivons plus loin sont :
• La composition fonctionnelle[8] (opération binaire) et le changement de type (opération unaire) dont l'inclusion dans la syntaxe des types a un précédent avec les travaux de Lambek (1958, 1961) et Geach (1972).
• La substitution fonctionnelle (opération binaire) proposée par Szabolcsi (1987).

Le calcul sur les types est donné par les règles d'inférence suivantes :

X/Y - Y ----------> X	;	Y - X\Y ----------< X
X/Y - Y/Z ----------**>B** X/Z	;	X/Y - Y\Z ----------**>Bx** X\Z
Y\Z - X\Y ----------**<B** X\Z	;	Y/Z - X\Y ----------**<Bx** X/Z
X ----------**>T** Y/(Y\X)	;	X ----------**>Tx** Y/(Y/X)
X ----------**<T** Y\(Y/X)	;	X ----------**<Tx** Y\(Y\X)
(X/Y)/Z - Y/Z ----------**>S** X/Z	;	(X/Y)\Z - Y\Z ----------**>Sx** X\Z
Y\Z - (X\Y)\Z ----------**<S** X\Z	;	Y/Z - (X\Y)/Z ----------**<Sx** X/Z

Dans le cadre de la Grammaire Catégorielle Combinatoire les catégories sont vues à la fois comme des objets syntaxiques et sémantiques. elles sont représentées par une structure de donnée informatique unique dans le but de réaliser une implémentation d'un analyseur, basée sur l'unification (Zeevat, Klein, Calder, 1986), (Pareschi, Steedman, 1987). Ainsi, à une analyse syntaxique, fondée sur un calcul inférentiel sur les types est associé un calcul fondé sur le lambda-calcul pour construire l'interprétation fonctionnelle. La construction de l'interprétation fonctionnelle est parallèle à la construction syntaxique.

---

[7] Ces nouvelles opérations permettent d'analyser des énoncés d'une façon quasi-incrémentale.

[8] L'introduction d'opérations fonctionnelles telle que la composition a l'effet de généraliser la notion de constituant de surface en ce sens que les verbes et les syntagmes verbaux ne sont pas les seuls constituants. Des séquences comme *peut écrire* et *Jean peut* le sont aussi. Prenons le cas de *peut écrire*, un verbe comme *pouvoir* est un foncteur qui prend comme argument un syntagme verbal, il est donc permis de le composer avec un autre foncteur ayant un type approprié comme *écrire* qui est une fonction de N dans S\N.

Dans cette optique la catégorie du verbe *aimer* est la suivante :

(S : aimer' np2 np1\NP : np1)/NP : np2

Avec cette notation (Steedman, 1989), les types syntaxiques sont en lettres majuscules, les constantes sémantiques portent des symboles ('), les lettres minuscules sont des variables sémantiques.

Chaque règle est en même temps syntaxique et sémantique. Les règles combinatoires que propose donc Steedman sont :

```
[X/Y : λ x (f x)] - [Y : a]                          [Y : a] - [X\Y : λ x (f x)]
---------------------------->                   ;    ----------------------------<
[X : (f a)]                                          [X : (f a)]
[X/Y : λ y (f y)] - [Y/Z : λ x (g x)]                 [X/Y : λ y (f y)] - [Y\Z : λ x (g x)]
----------------------------------------->B     ;    ----------------------------------------->Bx
[X/Z : λ x (f (g x))]                                [X\Z : λ x (f (g x))]
[Y\Z : λ x (g x)] - [X\Y : λ y (f y)]                 [Y/Z : λ x (g x)] - [X\Y : λ y (f y)]
----------------------------------------<B     ;    ----------------------------------------<Bx
[X\Z : λ x (f (g x))]                                [X/Z : λ x (f (g x))]
[X : x]                                              [X : x]
------------------------>T                     ;    ------------------------>Tx
[Y/(Y\X) : λ f (f x)]                                [Y/(Y/X) : λ f (f x)]
[X : x]                                              [X : x]
------------------------<T                     ;    ------------------------<Tx
[Y\(Y/X) : λ f (f x)]                                [Y\(Y\X) : λ f (f x)]
[(X/Y)/Z : λ x λ y (f x y)] - [Y/Z : λ x (g x)]       [(X/Y)\Z : λ x λ y (f x y)] - [Y\Z : λ x (g x)]
------------------------------------------------>S ;    ------------------------------------------------>Sx
[X/Z : λ x (f x (g x))]                              [X\Z : λ x (f x (g x))]
[Y\Z : λ x (g x)] - [(X\Y)\Z : λ x λ y (f x y)]       [Y/Z : λ x (g x)] - [(X\Y)/Z : λ x λ y (f x y)]
------------------------------------------------<S ;    ------------------------------------------------<Sx
[X\Z : λ x (f x (g x))]                              [X/Z : λ x (f x (g x))]
```

Aux règles et types syntaxiques proposés par Steedman, sont associées des interprétations sémantiques qui nous permettent de rendre compte de l'aspect fonctionnel des énoncés. Steedman propose un calcul fondé sur le lambda-calcul et l'unification pour construire l'interprétation sémantique.

```
Jean-                aime-                                  Marie
-----                -----                                  ------
N:Jean'              (S:aime' np2 np1\NP:np1)/NP:np2        NP:Marie'
--------->T
S:pred Jean'/(S:pred Jean'\NP:Jean')
-------------------------------------------------------------------------------->B
S:aime' np2 Jean'/NP:np2
-------------------------------------------------------------------------------------------->
S:aime' Marie' Jean'
```

L'effet immédiat de la règle de changement de type est de permettre au type fonctionnel obtenu à partir du type de l'argument *Jean* de se composer avec le type du prédicat *aime* ; ce qui déclenche une opération d'unification. Cette composition de types étant permise par les règles de composition fonctionnelle.

Les principaux objectifs visés par Steedman avec le modèle de la Grammaire Catégorielle Combinatoire sont d'une part éliminer le problème de la pseudo-ambiguïté par une stratégie d'analyse quasi-incrémentale de "gauche à droite" et d'autre part analyser des énoncés que le modèle de Ajdukiewicz-Bar-Hillel ne peut pas analyser.

3. LA GRAMMAIRE CATÉGORIELLE COMBINATOIRE APPLICATIVE

Comme nous l'avons annoncé au début de cet article, notre travail consiste à relier le phénotype au génotype à travers l'utilisation d'un système formel : la Grammaire Catégorielle Combinatoire Applicative (GCCA).

Nous considérons dans la GCCA que les règles de la Grammaire Catégorielle Combinatoire de Steedman introduisent les combinateurs[9] **B**, $\mathbf{C}_*$, **S** dans la séquence syntagmatique. Cette introduction permet de passer d'une structure concaténée à une structure applicative.

Plus concrètement, GCCA établit une association canonique entre les règles catégorielles combinatoires de Steedman et les combinateurs de la logique combinatoire de Curry. Les règles catégorielles auront pour rôle de vérifier la correction syntaxique des énoncés. L'introduction des combinateurs dans la chaîne syntagmatique dans un premier temps conduit à la construction des expressions applicatives dans un second temps avec la réduction des combinateurs. L'association des règles catégorielles avec les combinateurs atteste que les deux approches, en l'occurrence : la Grammaire Catégorielle Combinatoire de Steedman, les deux niveaux de représentations des langues proposés par S.K. Shaumyan et J.P. Desclés, peuvent être complémentaire. Par ailleurs, nous soutenons aussi le principe (trivial peut être) que la Grammaire Catégorielle Combinatoire agit sur des structures concaténées, alors que les combinateurs agissent sur des structures applicatives.

Les règles de la GCCA sont :

Règles d'application :

$[X/Y : u_1]\text{-}[Y : u_2]$		$[Y : u_1]\text{-}[X\backslash Y : u_2]$
---------->	;	----------<
$[X : (u_1\ u_2)]$		$[X : (u_2\ u_1)]$

Règles de changement de type :

$[X : u]$		$[X : u]$
----------**>T**	;	----------**<T**
$[Y/(Y\backslash X) : (\mathbf{C}_*\ u)]$		$[Y\backslash(Y/X) : (\mathbf{C}_*\ u)]$
$[X : u]$		$[X : u]$
----------**>Tx**	;	----------**<Tx**
$[Y/(Y/X) : (\mathbf{C}_*\ u)]$		$[Y\backslash(Y\backslash X) : (\mathbf{C}_*\ u)]$

Règles de composition fonctionnelle :

$[X/Y : u_1]\text{-}[Y/Z : u_2]$		$[Y\backslash Z : u_1]\text{-}[X\backslash Y : u_2]$
----------**>B**	;	----------**<B**
$[X/Z : (\mathbf{B}\ u_1\ u_2)]$		$[X\backslash Z : (\mathbf{B}\ u_2\ u_1)]$
$[X/Y : u_1]\text{-}[Y\backslash Z : u_2]$		$[Y/Z : u_1]\text{-}[X\backslash Y : u_2]$
----------**>Bx**	;	----------**<Bx**
$[X\backslash Z : (\mathbf{B}\ u_1\ u_2)]$		$[X/Z : (\mathbf{B}\ u_2\ u_1)]$

---

[9] Voir en annexe la notion de combinateur.

Règles de composition "distributive" :

[(X/Y)/Z : u_1]-[Y/Z : u_2]		[Y\Z : u_1]-[(X\Y)\Z : u_2]
----------->**S**	;	----------<**S**
[X/Z : (**S** u_1 u_2)]		[X\Z : (**S** u_2 u_1)]
[X/Y)\Z : u_1]-[Y\Z : u_2]		[Y/Z : u_1]-[(X\Y)/Z : u_2]
----------->**Sx**	;	----------<**Sx**
[X\Z : (**S** u_1 u_2)]		[X/Z : (**S** u_2 u_1)]

Les prémisses des règles sont des *expressions concaténées typées* ; les résultats sont des *expressions applicatives (typées) avec éventuellement introduction d'un combinateur.* Le changement de type d'une unité u introduit le combinateur $\mathbf{C}_*$; la composition de deux unités concaténées introduit le combinateur **B** et **S**. Avec ces règles nous pouvons analyser une phrase au moyen d'une stratégie quasi-incrémentale "de gauche à droite". Le choix d'une telle stratégie est motivé par le contrôle du problème de la pseudo-ambiguïté (Pareschi, Steedman, 1987) (Steedman, 1989).

Exemple :

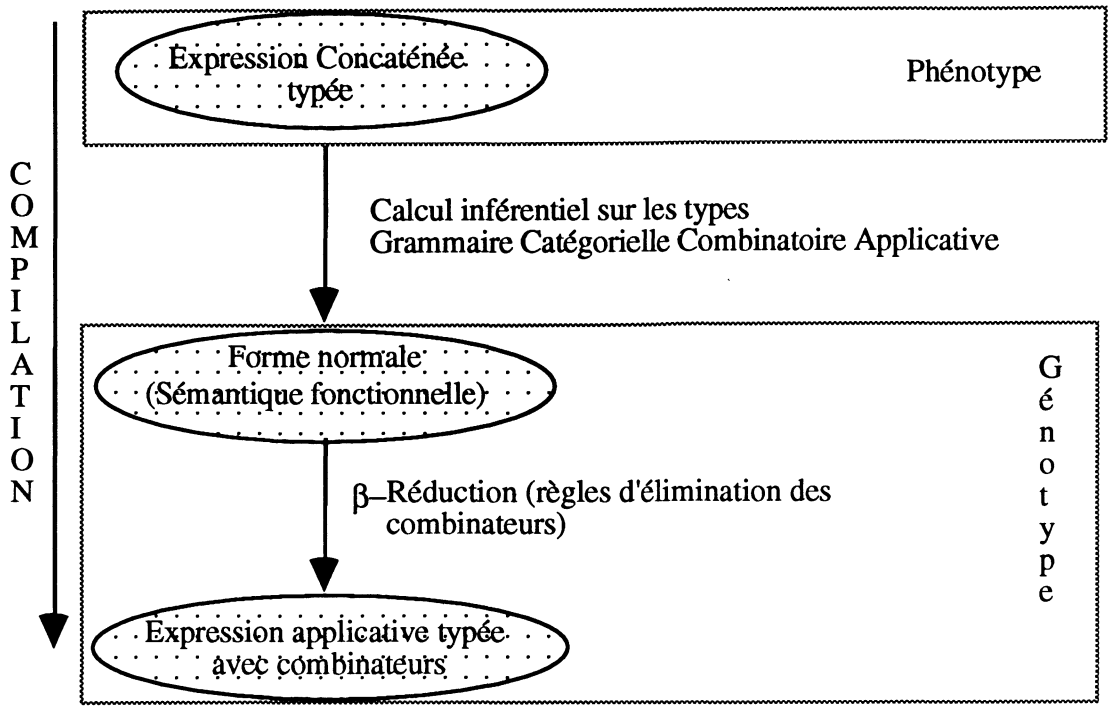
Jean	-	*aime*
------		------
[N : *Jean*]		
----------->**T**		
[S/(S\N) : ($\mathbf{C}_*$ *Jean*)]	-	[(S\N)/N : *aime*]
----------->**B**		
[S/N : (**B** ($\mathbf{C}_*$ *Jean*) *aime*)]		

La première règle (**>T**) appliquée à l'unité typée [N : *Jean*] transforme l'opérande en opérateur. Elle construit une structure applicative ($\mathbf{C}_*$ *Jean*) ayant pour type S/(S\N). L'introduction du combinateur $\mathbf{C}_*$ illustre dans la représentation applicative le changement de type : ($\mathbf{C}_*$ *Jean*) fonctionne comme un opérateur avec son type fonctionnel. La règle (**>B**) combinent les unités linguistiques typées [S/(S\N) : ($\mathbf{C}_*$ *Jean*)] et [(S\N)/N : *aime*] avec le combinateur **B** de façon à pouvoir composer les deux unités fonctionnelles ($\mathbf{C}_*$ *Jean*) et *aime.*

Un traitement complet basé sur la Grammaire Catégorielle Combinatoire Applicative s'effectue en deux grandes étapes :

(i) la première étape s'illustre par la vérification de la bonne connexion syntaxique et la construction de structures fonctionnelles avec des combinateurs introduits à certaines positions de la chaîne syntagmatique,
(ii) la deuxième étape consiste à utiliser les règles de β–réduction des combinateurs de façon à former une structure fonctionnelle sous-jacente à l'expression phénotypique. L'expression obtenue est applicative et appartient au langage génotype. La GCCA engendre des processus qui associent une structure applicative à une expression concaténée du phénotype. Il nous reste à éliminer les combinateurs de l'expression obtenue de façon à construire la "forme normale" (au sens technique de la β–réduction) qui exprime l'*interprétation sémantique fonctionnelle.* Ce calcul s'effectue entièrement dans le génotype.

Le traitement que nous proposons donc prend la forme d'une "compilation" dont les étapes sont résumées dans la figure ci-dessous :



Traitons un exemple simple : *Jean aime Marie*

1	[N:*Jean*]-[(S\N)/N:*aime*]-[N:*Marie*]	**Structure concaténée typée du phénotype**
2	[S/(S\N):C_* *Jean*)]-[S\N)/N:*aime*]-[N:*Marie*]	(**>T**)
3	[S/N:(**B**(C_**Jean*) *aime*)]-[N:*Marie*]	(**>B**) **Compilation**
4	[S:((**B** (C_* *Jean*) *aime*) *Marie*)]	(**>**)
		V
5	[S : ((**B** (C_* *Jean*) *aime*) *Marie*)]	**Structure applicative typée du génotype**
6	[S : ((C_* *Jean*) (*aime Marie*))]	(**B**)
7	[S : ((*aime Marie*) *Jean*)]	(C_*)
		Forme normale du génotype

Le changement de type (**>T**) affectant l'opérande *Jean* permet d'engendrer l'opérateur (C_* *Jean*) que la règle fonctionnelle (**>B**) compose avec l'opérateur *aime*. L'opérateur complexe (**B** (C_* *Jean*) *aime*) s'applique à l'opérande *Marie* pour former l'expression applicative du génotype ((**B** (C_* *Jean*) *aime*) *Marie*). La réduction des combinateurs dans le génotype construit l'interprétation sémantique fonctionnelle sous-jacente à l'expression phénotypique en entrée.

3.1. La réorganisation structurelle

L'analyse syntaxique "de gauche à droite" soulève le problème du non-déterminisme introduit par la présence dans la langue, de modifieurs arrières qui sont des opérateurs qui s'appliquent à l'ensemble ou une partie d'une structure préalablement construite.

Si dans le premier cas l'utilisation d'une règle d'application permet la poursuite de l'analyse[10], il en est autrement pour le second cas où l'analyse "patine". Pour une phrase comme *Jean aime Marie tendrement* l'analyseur produit dans un premier temps le constituant [S : ((**B**(**C**$_*$ *Jean*) *aime*) *Marie*)]. Ce dernier n'est pas combinable avec *tendrement*, de type (S\N)\(S\N). En effet, *tendrement* est un opérateur qui a comme opérande (*aime Marie*) positionné à sa gauche. Une analyse quasi-incrémentale de "gauche à droite" favorise l'application d'une règle combinatoire dès que possible. Ce facteur a pour conséquence directe d' "enchâsser"[11] *aime* et *Marie* dans ((**B** (**C**$_*$ *Jean*) *aime*) *Marie*), ce qui évidemment ne nous permet pas de construire directement l'opérande (*aime Marie*).

Le problème posé revient à la possibilité d'un retour arrière. Mais ce retour arrière est de nature à accroître le coût "computationnel" (mémoire et temps d'exécution) d'une analyse syntaxique. Cependant, un retour arrière "intelligent" (que nous allons proposer plus loin) peut nous permettre de réduire considérablement ce coût tout en construisant des analyses sémantiques correctes et en éliminant les pseudo-ambiguïtés.

Ainsi un tel retour arrière décomposera le constituant déjà construit en deux composantes dont une se combine forcément avec le modifieur arrière.

Formellement, cette opération de réorganisation structurelle s'effectue par les deux étapes successives suivantes:

a - La réorganisation du constituant déjà construit isole à chaque fois deux sous-catégories, et teste si le modifieur arrière se combine à gauche[12], ou pas, avec une de ces deux sous-catégories. Nous procédons ensuite à la réduction des combinateurs jusqu'à ce que le test nous donne une valeur positive. A la fin du processus nous récupérons une nouvelle structure applicative typée "équivalente" à la première.

Exemple. Dans le cas de l'énoncé *Jean aime Marie tendrement*, les étapes de la réorganisation sont :

Le constituant construit : [S : ((**B** (**C**$_*$ *Jean*) *aime*) *Marie*)]

Les deux sous-catégories sont : [S/N : (**B** (**C**$_*$ *Jean*) *aime*)] ; [N : *Marie*]

Test : [S/N : (**B** (**C**$_*$ *Jean*) *aime*)] ne se combine pas à gauche avec [(S\N)\(S\N) : *tendrement*]

[N : *Marie*] ne se combine pas à gauche avec [(S\N)\(S\N) : *tendrement*]

Réduction du combinateur **B** : [S : ((**C**$_*$ *Jean*) (*aime Marie*))]

Les deux sous-catégories sont : [S/(S\N) : (**C**$_*$ *Jean*)] ; [S\N : (*aime Marie*)]

---

[10] Prenons l'exemple de la phrase *Jean frappa Marie hier* où le modifieur arrière *hier* opère sur l'ensemble de la phrase *Jean frappa Marie* ; *hier* étant de type syntaxique S\S, il suffit, pour poursuivre l'analyse, d'appliquer *hier* à *Jean frappa Marie* par la règle (<).

[11] C'est à dire que *Marie* n'apparait pas comme l'opérande directe de l'opérateur *aimer*.

[12] Dans notre terminologie, u_1 se combine à gauche avec u_2 si une des règles suivantes **<**, **<B**, **<Bx**, **<S**, **<Sx**, peut composer les types de u_2 et u_1 ou si une des règles combinatoires peut composer les types de (**C**$_*$ u_2) et u_1.

Test : [S/(S\N) : ($\mathbf{C}_*$ *Jean*)] ne se combine pas à gauche avec
[(S\N)\(S\N) : *tendrement*]

[S\N : (*aime Marie*)] se combine à gauche avec
[(S\N)\(S\N) : *tendrement*]

Arrêt du processus de réduction des combinateurs. Nous récupérons en sortie la catégorie:

[S : (($\mathbf{C}_*$ *Jean*) (*aime Marie*))].

b - La décomposition réalisée grâce aux deux règles[13] :

$$\frac{[X : (u_1\ u_2)]}{[X/Y : u_1]\text{-}[Y : u_2]}\ \mathbf{>dec} \quad ; \quad \frac{[X : (u_1\ u_2)]}{[Y : u_2]\text{-}[X\backslash Y : u_1]}\ \mathbf{<dec}$$

Nous lisons ces règles comme suit :

- Pour (**>dec**): Si nous avons une structure applicative (u_1 u_2) de type X, avec u_1 de type X/Y et u_2 de type Y, alors nous pouvons construire une nouvelle expression concaténée formée des deux catégories [X/Y : u_1] et [Y : u_2].

- Pour (**<dec**): Si nous avons une structure applicative (u_1 u_2) de type X, avec u_1 de type X\Y et u_2 de type Y, alors nous pouvons construire une nouvelle expression concaténée formée des deux catégories [Y : u_2] et [X\Y : u_1].

Ces deux règles nous permettent de reconstruire un nouvel agencement concaténé de la structure opérateur/opérande issue de la réorganisation.

Pour la phrase *Jean aime Marie tendrement* la décomposition est appliquée à la structure qui résulte de la réorganisation :

[S : (($\mathbf{C}_*$ *Jean*) (*aime Marie*))]

Avec la règle (**>dec**), nous produisons l'agencement concaténé :

[S/(S\N) : ($\mathbf{C}_*$ *Jean*)] - [S\N : (*aime Marie*)].

Ces deux étapes entrent dans l'analyse complète de la phrase *Jean aime Marie tendrement* comme suit (étape 5 pour la réorganisation et étape 6 pour la décomposition) :

---

[13] Notons que les deux règles (**>dec**) et (**<dec**) sont respectivement inverses des règles d'application fonctionnelle (>) et (<).

Structure concaténée typée du phénotype

1	[N : *Jean*]-[(S\N)/N : *aime*]-[N : *Marie*]-[(S\N)\(S\N) : *tendrement*]	
...		
4	[S : ((**B** (C_* *Jean*) *aime*) *Marie*)]-[(S\N)\(S\N) : *tendrement*]	
5	[S : ((C_* *Jean*) (*aime Marie*))]-[(S\N)\(S\N) : *tendrement*]	**(B)**
6	[S/(S\N) : (C_* *Jean*)]-[S\N : (*aime Marie*)]-[(S\N)\(S\N) : *tendrement*]	**(dec>)**
7	[S/(S\N) : (C_* *Jean*)]-[S\N : (*tendrement* (*aime Marie*))]	**(<)**
8	[S : ((C_* *Jean*) (*tendrement* (*aime Marie*)))]	**(>)**

Structure applicative typée du génotype

9	[S : ((C_* *Jean*) (*tendrement* (*aime Marie*)))]		**Forme normale du génotype**
10	[S : ((*tendrement* (*aime Marie*)) *Jean*)]	(C_*)	

3.2. La coordination

La coordination est l'action de joindre deux mots ou deux expressions du même genre ou ayant même fonction. Dans le cadre des Grammaires Catégorielles, Steedman (1989), Barry et Pickering (1990) considèrent que deux unités linguistiques peuvent être coordonnées pour donner une unité linguistique de type X si et seulement si chaque unité est de type X. Même si cette définition reste incomplète, sachant que la coordination se présente sous différentes formes, elle indique de manière idéale la voie à suivre pour établir une solution fiable.

Présentons quatre types d'exemples de coordination par ET. Nous pouvons coordonner[14] :

a) Deux segments de même type, de même structure et contigus à ET :

[*Jean aime*]$_{S/N}$ *et* [*Paul déteste*]$_{S/N}$ *ces tableaux*

b) Deux segments dans une construction elliptique :

Jean aime [*Marie tendrement*] *et* [*Sophie sauvagement*]

[*Jean*] *aime* [*Marie*] *et* [*Paul Sophie*]

c) Deux segments de structures différentes :

Marie marche [*doucement*] *et* [*avec élégance*].

Jean [*chante*] *et* [*joue du violon*].

d) Deux segments, sans distributivité:

Le drapeau est [*blanc*] *et* [*rouge*]
(≠ *Le drapeau est blanc et le drapeau est rouge*).

---

[14] Les catégories à coordonner sont entre crochets.

A la conjonction 'ET' on associe le type polymorphique (X\X)/X[15]. Le contexte donne cependant plus de spécifications pour assigner un type à 'ET'.

Les hypothèses 1 et 2 permettent d'assigner un type à ET en tenant compte du contexte.

HYPOTHÈSE 1. *La catégorie construite qui suit immédiatement la conjonction ET détermine le type de la coordination.*

Cette hypothèse nous amène indirectement à introduire une interruption dans l'analyse quasi-incrémentale : dès que nous rencontrons la conjonction ET nous interrompons momentanément l'analyse quasi-incrémentale pour construire le second membre de la coordination.

Nous revenons ensuite en arrière pour déterminer le premier membre de la coordination.

Nous proposons la seconde hypothèse:

HYPOTHÈSE 2. *Quand nous avons une coordination de type X définie par l'hypothèse 1, le premier membre de la coordination est la catégorie de type X qui précède immédiatement la conjonction.*

Les règles que nous devons dégager à travers ces deux hypothèses partent donc de l'idée que les deux membres de la coordination ont des types syntaxiques X identiques correspondant à des interprétations sémantiques fonctionnelles différentes. Le résultat de l'application de ces règles conserve le même type syntaxique X. Nous établissons deux types abstraits pour la conjonction. Le premier concerne la conjonction distributive, nous le noterons CONJD. Le deuxième type concerne la conjonction non distributive et nous le noterons CONJN.

$$\frac{[X : u_1]\text{-}[CONJD : et]\text{-}[X : u_2]}{[X : (\Phi \text{ et } u_1\ u_2)]}\ \textbf{<CONJD>} \quad ; \quad \frac{[X : u_1]\text{-}[CONJN : et]\text{-}[X : u_2]}{[X : (\text{et } u_1\ u_2)]}\ \textbf{<CONJN>}$$

Nous appliquons la règle **<CONJD>** aux cas de la coordination distributive. Pour prendre en compte la distributivité au niveau de la structure applicative, nous utilisons le combinateur Φ. Nous appliquons la règle **<CONJN>** aux cas de la coordination non distributive (voir exemple E3).

Avec l'analyse quasi-incrémentale, lors de l'application de l'hypothèse 2, deux cas de figure se présentent :

- Le constituant produit avant de rencontrer la conjonction est du même type que le constituant déterminé par la coordination. Ce constituant est alors le premier membre de la coordination. Par exemple, l'analyse de la phrase : [*Jean aime*]$_{S/N}$ *et* [*Paul déteste*]$_{S/N}$ *ces tableaux* construit [S/N : (**B**(**C**$_*$*Jean*) *aime*)] avant de rencontrer la conjonction. Le type est identique au type du second membre [S/N : (**B** (**C**$_*$*Paul*) *déteste*)], le constituant déterminé par la première hypothèse. Le constituant [S/N : (**B**(**C**$_*$*Jean*) *aime*)] est alors le premier membre de la coordination.

---

[15] (Lambek, 1958).

- Le constituant déterminé avant de rencontrer la conjonction n'est pas du même type que le constituant déterminé par la coordination. Il est nécessaire de modifier la structure de ce constituant. Par exemple, l'analyse de la phrase : *Jean aime [Marie tendrement] et [Sophie sauvagement]* construit [S : (($\mathbf{C}_*$ *Jean*) (*tendrement* (*aime Marie*)))] avant l'analyse de la conjonction. Le second membre de la coordination étant [(S\N)\((S\N)/N) : (**B** sauvagement ($\mathbf{C}_*$ Sophie))][16].

Dans ce second cas le processus de réorganisation structurelle nous permet :

- soit d'isoler directement le premier membre de la coordination[17].

- soit d'isoler la structure binaire "opérateur/opérande" qui contient le premier membre de la coordination. Dans cette deuxième situation, il est nécessaire d'adjoindre à la réorganisation structurelle l'utilisation des équivalences logiques[18] de la logique combinatoire (e,f,g,h)[19] :

(e)	(u1 (u2 u3))	<==>	((**B** u1 u2) u3)
(f)	((u1 u2) u3)	<==>	((**B** ($\mathbf{C}_*$ u3) u1) u2)
(g)	(u1 (u2 u3))	<==>	((**B** u1 ($\mathbf{C}_*$ u3)) u2)
(h)	((u1 u2) u3)	<==>	((**B** ($\mathbf{C}_*$ u3) ($\mathbf{C}_*$ u2)) u1)

3.3. Les métarègles

Nous enrichissons notre formalisme par différentes métarègles qui contrôlent le changement de type.

Ces métarègles d'une part nous indiquent qu'une règle de changement de type doit être appliquée, et d'autre part choisissent le changement de type particulier à effectuer.

Nous ne concevons pas ces métarègles comme un outil purement informatique, nous leur donnons une pertinence logique et linguistique. Elles peuvent avoir une interprétation si nous tenons compte des facteurs prosodiques.

Soient u_1 et u_2 dans l'expression concaténée 'u_1-u_2' :

MÉTARÈGLE 1. Si u_1 est de type N et u_2 de type (Y\N)/Z, alors nous appliquons le changement de type avant (**>T**) à u_1 :

[N : u_1 ==> Y/(Y\N) : ($\mathbf{C}_*$ u_1)]

Exemple: *Jean mange la pomme*

[N : *Jean*]-[(S\N)/N : *mange*]-[N/N : *la*]-[N : *pomme*]
[S/(S\N) : ($\mathbf{C}_*$ *Jean*)]-[(S\N)/N : *mange*]-[N/N : *la*]-[N : *pomme*]

Dans ce cas Y = S ; Z = N.

---

[16] L'énoncé *Jean aime Marie tendrement et Sophie sauvagement* est ambigu. Dans notre exemple, nous considérons que *Sophie* est objet.

[17] Voir les étapes 6 et 7 de l'exemple E1.

[18] Ces équivalences sont les conséquences directes de l'introduction et l'élimination des combinateurs **B** et $\mathbf{C}_*$.

[19] Voir l'étape 8 de l'exemple E2

MÉTARÈGLE 2. Si u1 est de type X et u2 de type (Y\X)/Z, alors nous appliquons le changement de type avant à u1 (**>T**) :

[X : u1 ==> Y/(Y\X) : (**C**$_*$ u1)]

Exemple :*Jean court très lentement*

[N:*Jean*]-[S\N:*court*]-[((S\N)\(S\N))/((S\N)\(S\N)):*très*]-[(S\N)\(S\N)):*lentement*]
[N:*Jean*]-[(S\N)/((S\N)\(S\N)):(**C**$_*$*court*)]-[((S\N)\(S\N))/((S\N)\(S\N)):*très*]-[(S\N)\(S\N)):*lentement*]

Dans ce cas X = S\N ; Y = S\N ; Z = (S\N)\(S\N).

MÉTARÈGLE 3. Si u2 est de type N et u1 de type (Y/N)\X, alors nous appliquons le changement de type arrière (**<T**) à u2:

[N : u2 ==> Y\(Y/N) : (**C**$_*$ u2)]

Exemple :*Jean aime tendrement Marie et sauvagement Sophie*

...-[CONJD : *et*]-[((S\N)/N)\((S\N)/N):*sauvagement*]-[N : *Sophie*]
...-[CONJD: *et*]-[(S\N)/N)\((S\N)/N):*sauvagement*]-[(S\N)\((S\N)/N): (**C**$_*$ *Sophie*)]

Dans ce cas X = ((S\N)/N) ; Y = S\N.

MÉTARÈGLE 4. Si u2 est de type N et u1 de type Y/S, alors nous appliquons le changement de type avant (**>T**) à u2 :

[N : u2 ==> S/(S\N) : (**C**$_*$ u2)]

Exemple :*Jean voit Paul tuer Marie*

[S/S : (**B** (**C**$_*$ *Jean*) *voit*)]-[N : *Paul*]-...
[S/S : (**B** (**C**$_*$ *Jean*) *voit*)]-[S/(S\N) : (**C**$_*$ *Paul*)]-...

Dans ce cas Y = S.

MÉTARÈGLE 5. Si u1 est de type N (u1 précédé de *et*) et u2 de type N, alors nous appliquons le changement de type avant (**>T**) à u1 :

[N : u1 ==> S/(S\N) : (**C**$_*$ u1)]

Exemple :*Jean aime Marie et Paul Sophie*

...-[CONJD : *et*]-[N : *Paul*]-[N : *Sophie*]
...-[CONJD : *et*]-[S/(S\N) : (**C**$_*$ *Paul*)]-[N : *Sophie*]

MÉTARÈGLE 6. Si u2 est de type N et u1 de type Y/X (u1 précédé de *et*), alors nous appliquons le changement de type arrière (**<T**) à u2 :

[N : u2 ==> X\(X/N) : (**C**$_*$ u2)]

Exemple :*Jean aime Marie et Paul Sophie*

...-[CONJD : *et*]-[S/(S\N) : (C_* *Paul*)]-[N : *Sophie*]
...-[CONJD : *et*]-[S/(S\N) : (C_* *Paul*)]-[(S\N)\((S\N)/N) :(C_* *Sophie*)]

Dans ce cas X = S\N ; Y = S.

MÉTARÈGLE 7. Si u1 est de type N (u1 précédé de *et*) et u2 de type Y\X, alors nous appliquons le changement de type arrière (**<T**) à u1:

[N : u1 ==> X\(X/N) : (C_* u1)]

Exemple : *Jean aime Marie tendrement et Sophie sauvagement.*

...-[CONJD : *et*]-[N : *Sophie*]-[(S\N)\(S\N) : *sauvagement*]
...-[CONJD : *et*]-[(S\N)\((S\N)/N) : (C_* *Sophie*)]-[(S\N)\(S\N) : *sauvagement*]

Dans ce cas X = S\N ; Y = S\N.

MÉTARÈGLE 8. Si u1 est de type N (u1 précédé de *et*) et u2 de type (X\N)\Y, alors nous appliquons le changement de type avant (**>T**) à u1:

[N : u1 ==> X/(X\N) : (C_* u1)]

Exemple : *Jean aime Marie tendrement et Sophie sauvagement.*

...-[CONJD : *et*]-[N : *Sophie*]-[(S\N)\(S\N) : *sauvagement*]
...-[CONJD : *et*]-[S/(S\N) : (C_* *Sophie*)]-[(S\N)\(S\N) : *sauvagement*]

Dans ce cas X = S ; Y = S\N.

MÉTARÈGLE 9. Soit le schéma suivant : "u1-et-u2-u3" (*et* de type CONJD). Si u1 et u2 sont de type N, et u3 de type (S\N)/X, alors nous appliquons le changement de type avant (**>T**) à u1 et u2 :

[N : u1 ==> S/(S\N) : (C_* u1)]
[N : u2 ==> S/(S\N) : (C_* u2)]

Exemple :*Jean et Paul aiment Marie*

[N : *Jean*]-[CONJD : *et*]-[N : *Paul*]-[(S\N)/N : *aiment*]-[N : *Marie*]
[S/(S\N):(C_**Jean*)]-[CONJD:*et*]-[S/(S\N):(C_**Paul*)]-[(S\N)/N:*aiment*]-[N:*Marie*]

MÉTARÈGLE 10. Soit le schéma suivant "u1-u2-et-u3" (*et* de type CONJD). Si u2 et u3 sont de type N, et u1 of type Y/N, alors nous appliquons le changement de type arrière (**<T**) à u2 et u3 :

[N : u2 ==> (S\N)\((S\N)/N) : (C_* u2)]
[N : u3 ==> (S\N)\((S\N)/N) : (C_* u3)]

Exemple : *Jean aime Marie et Sophie*

[N : *Jean*]-[(S\N)/N : *aime*]-[N : *Marie*]-[CONJD : *et*]-[N : *Sophie*]
[N:*Jean*]-[(S\N)/N:*aime*]-[(S\N)\((S\N)/N):(C_**Marie*)]-[CONJD:*et*]-[(S\N)\((S\N)/N):(C_**Sophie*)]

Les métarègles 1 à 8 sont utilisées pour construire des types syntaxiques qui se combineront avec l'utilisation des règles d'application, de composition, de composition distributive. Les métarègles 9 et 10 sont nécessaires pour l'analyse de la coordination distributive de deux opérandes. Le combinateur **Φ** qui exprime la distributivité s'appliquera ainsi à deux opérateurs.

3.4. Exemples .

E1 : *Jean aime Marie et déteste Sophie*

Phénotype

1	[N:*Jean*]-[(S\N)/N:*aime*]-[N:*Marie*]-[CONJD:*et*]-[(S\N)/N:*déteste*]-[N:*Sophie*]	
...		
4	[S:((**B** (**C**$_*$ *Jean*) *aime*) *Marie*)]-[CONJD:*et*]-[(S\N)/N:*déteste*]-[N:*Sophie*]	
5	[S:((**B** (**C**$_*$ *Jean*) *aime*) *Marie*)]-[CONJD:*et*]-[S\N:(*déteste Sophie*)]	**(>)**
6	[S:((**C**$_*$ *Jean*) (*aime Marie*))]-[CONJD:*et*]-[S\N:(*déteste Sophie*)]	**(B)**
7	[S/(S\N):(**C**$_*$ *Jean*)]-[S\N:(*aime Marie*)]-[CONJD:*et*]-[S\N:(*déteste Sophie*)]	**(>dec)**
8	[S/(S\N):(**C**$_*$ *Jean*)]-[S\N:(**Φ** *et* (*aime Marie*) (*déteste Sophie*))]	**(<CONJD>)**
9	[S:((**C**$_*$ *Jean*) (**Φ** *et* (*aime Marie*) (*déteste Sophie*)))]	**(>)**

Génotype

10	[S : ((**C**$_*$ *Jean*) (**Φ** *et* (*aime Marie*) (*déteste Sophie*)))]	
11	[S : ((**Φ** *et* (*aime Marie*) (*déteste Sophie*)) *Jean*)]	**(C**$_*$**)**
12	[S : (*et* ((*aime Marie*) *Jean*) ((*déteste Sophie*) *Jean*))]	**(Φ)**

E2 : *Jean aime Marie et Paul Sophie*

Phénotype

1	[N:*Jean*]-[(S\N)/N:*aime*]-[N:*Marie*]-[CONJD:*et*]-[N:*Paul*]-[N:*Sophie*]	
...		
4	[S:((**B** (**C**$_*$ *Jean*) *aime*) *Marie*)]-[CONJD:*et*]-[N:*Paul*]-[N:*Sophie*]	
5	...-[CONJD:*et*]-[S/(S\N):(**C**$_*$ *Paul*)]-[N:*Sophie*]	**(>T), M5**
6	...-[CONJD:*et*]-[S/(S\N):(**C**$_*$ *Paul*)]-[(S\N)\(S/(S\N)):(**C**$_*$ *Sophie*)]	**(<T), M6**
7	...-[CONJD:*et*]-[S\(S/(S\N)):(**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*)]	**(>Bx)**
8	[S:((**B** (**C**$_*$ *Jean*) (**C**$_*$ *Marie*)) *aime*)]-[CONJD:*et*]-...	**(h)**
9	[(S\N)/N:*aime*]-[S\(S/(S\N)):(**B** (**C**$_*$ *Jean*) (**C**$_*$ *Marie*))]-[CONJD:*et*]-...	**(<dec)**
10	...-[S\(S/(S\N)):(**Φ** *et* (**B**(**C**$_*$ *Jean*)(**C**$_*$ *Marie*))(**B**(**C**$_*$ *Paul*)(**C**$_*$ *Sophie*))]	**(<CONJD>)**
11	[S:((**Φ** *et* (**B** (**C**$_*$ *Jean*) (**C**$_*$ *Marie*))(**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*))) *aime*)]	**(<)**

Génotype

12	[S : ((Φ *et* (**B** (**C**$_*$ *Jean*) (**C**$_*$ *Marie*))(**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*))) *aime*)]	
13	[S : (*et* ((**B** (**C**$_*$ *Jean*) (**C**$_*$ *Marie*)) *aime*) ((**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*)) *aime*))]	(Φ)
14	[S : (*et* ((**C**$_*$ *Jean*) ((**C**$_*$ *Marie*) *aime*)) ((**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*)) *aime*))]	(**B**)
15	[S : (*et* (((**C**$_*$ *Marie*) *aime*) *Jean*) ((**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*)) *aime*))]	(**C**$_*$)
16	[S : (*et* ((*aime Marie*) *Jean*) ((**B** (**C**$_*$ *Paul*)(**C**$_*$ *Sophie*)) *aime*))]	(**C**$_*$)
17	[S : (*et* ((*aime Marie*) *Jean*) ((**C**$_*$ *Paul*)((**C**$_*$ *Sophie*) *aime*)))]	(**B**)
18	[S : (*et* ((*aime Marie*) *Jean*) (((**C**$_*$ *Sophie*) *aime*) *Paul*))]	(**C**$_*$)
19	[S : (*et* ((*aime Marie*) *Jean*) ((*aime Sophie*) *Paul*))]	(**C**$_*$)

E3 : *le drapeau est blanc et rouge*
=≠> *le drapeau est blanc et le drapeau est rouge*

Phénotype

1	[N/N:*le*]-[N:*drapeau*]-[(S\N)/(N\N):*est*]-[N\N:*blanc*]-[CONJN:*et*]-[N\N:*rouge*]	
2	[N:(*le drapeau*)]-[(S\N)/(N\N):*est*]-[N\N:*blanc*]-[CONJN:*et*]-[N\N:*rouge*]	
3	[S/(S\N):(**C**$_*$ (*le drapeau*))]-[(S\N)/(N\N):*est*]-[N\N:*blanc*]-[CONJN:*et*]-...	(**>T**), **M1**
4	[S/(N/N):(**B** (**C**$_*$ (*le drapeau*)) *est*)]-[N\N:*blanc*]-[CONJN:*et*]-[N\N:*rouge*]	(**>B**)
5	[S:((**B** (**C**$_*$ (*le drapeau*)) *est*) *blanc*)]-[CONJN:*et*]-[N\N:*rouge*]	(**>**)
6	[S/(N/N):(**B** (**C**$_*$ (*le drapeau*)) *est*)]-[N\N:*blanc*]-[CONJN:*et*]-[N\N:*rouge*]	(**>dec**)
7	[S/(N/N):(**B** (**C**$_*$ (*le drapeau*)) *est*)]-[N\N:(*et blanc rouge*)]	(**<CONJN>**)
8	[S:((**B** (**C**$_*$ (*le drapeau*)) *est*) (*et blanc rouge*))]	(**>**)

Génotype

9	[S : ((**B** (**C**$_*$ (*le drapeau*)) *est*) (*et blanc rouge*))]	
10	[S : ((**C**$_*$ (*le drapeau*)) (*est* (*et blanc rouge*)))]	(**B**)
11	[S : ((*est* (*et blanc rouge*)) (*le drapeau*))]	(**C**$_*$)

E4 : *Tous les garçons aiment Marie*[20]

Phénotype

1	[(S/(S\N))/N : *tous*]-[N/N : *les*]-[N : *garçons*]-[(S\N)/N : *aiment*]-[N : *Marie*]	
2	[(S/(S\N) : (**B** *tous les*)]-[N : *garçons*]-[(S\N)/N : *aiment*]-[N : *Marie*]	(>**B**)
3	[(S/(S\N))/N : *tous*]-[N/N : *les*]-[N : *garçons*]-[(S\N)/N : *aiment*]-[N : *Marie*]	
4	[(S/(S\N))/N : *tous*]-[N : (*les garçons*)]-[(S\N)/N : *aiment*]-[N : *Marie*]	(>)
5	[(S/(S\N) : (*tous* (*les garçons*))]-[(S\N)/N : *aiment*]-[N : *Marie*]	(>)
6	[S/N : (**B** (*tous* (*les garçons*)) *aiment*)]-[N : *Marie*]	(>**B**)
7	[S : ((**B** (*tous* (*les garçons*)) *aiment*) *Marie*)]	(>)

Génotype

8	[S : ((**B** (*tous* (*les garçons*)) *aiment*) *Marie*)]	
9	[S : ((*tous* (*les garçons*)) (*aiment Marie*))]	**B**

E5 : *Tout garçon aime une fille*

Phénotype

1	[(S/(S\N))/N : *tout*]-[N : *garçon*]-[(S\N)/N : *aiment*]-[(S\N)\(((S\N)/N)/N) : *une*]-[N : *fille*]	
2	[(S/(S\N) : (*tout garçon*)]-[(S\N)/N : *aiment*]-[(S\N)\(((S\N)/N)/N) : *une*]-[N : *fille*]	(>)
3	[(S/N : (**B** (*tout garçon*) *aiment*)]-[(S\N)\(((S\N)/N)/N) : *une*]-[N : *fille*]	(>**B**)
4	[(S/N : (**B** (*tout garçon*) *aiment*)]-[(S\N)\((S\N)/N) : (*une fille*)]	(>)
5	[(S/(S\N) : (*tout garçon*)]-[(S\N)/N : *aiment*]-[(S\N)\((S\N)/N) : (*une fille*)]	
6	[(S/(S\N) : (*tout garçon*)]-[S\N : ((*une fille*) *aiment*)]	(>)
7	[S : ((*tout garçon*)) ((*une fille*) *aiment*))]	(>)

Génotype

8	[S : ((*tout* (*les garçon*) ((*une fille*) *aiment*))]	

Les deux derniers exemples introduisent des quantifications. L'interprétation formelle adéquate de l'énoncé 4 est $(\forall x)$ ((garçon x) $\supset$ (aime Marie x)), alors que l'interprétation formelle de l'énoncé 5 est $(\forall x)$ $(\exists y)$ ((garçon x) $\supset$ ((fille y) $\wedge$ (aime y x))).

D'autres exemples et plus de détails sont donnés dans (Biskri, 1995). Les analyses sont implémentées. Nous ne reprenons pas ici les détails de l'algorithme.

---

[20] Dans une première approximation nous assignerons au déterminant *les* le type syntaxique N/N.

4. COMMENTAIRES

Comme le lecteur a du le remarquer notre approche utilise des outils purement formels pour réaliser une analyse syntaxique des expressions phénotypiques et construire leur interprétation sémantique fonctionnelle. L'utilisation des combinateurs nous permet de construire les interprétations sémantiques fonctionnelles sans faire appel à l'unification comme Steedman, Pareschi, et d'autres le font. Bien sûr, nous ne dénigrons pas l'unification en tant qu'opération. Celle-ci a bien fait ses preuves dans d'autres domaines. Seulement, nous pensons que son utilisation dans le formalisme de Steedman est très critiquable au vu du traitement de certains exemples. Nous avons ainsi à formuler un reproche principal :

La construction des interprétations sémantiques fonctionnelles fondée sur l'unification ne nous permet pas de garder une trace des opérations faites. Dans le cas de la GCCA, la trace est contenue dans les combinateurs eux mêmes. Le combinateur nous renseigne sur la règle utilisée. Prenons l'exemple du traitement de l'énoncé *Jean aime Marie* :

1	[N:*Jean*]-[(S\N)/N:*aime*]-[N:*Marie*]		
2	[S/(S\N):(**C**$_*$ *Jean*)]-[(S\N)/N:*aime*]-[N:*Marie*]	(**>T**)	
3	[S/N:(**B** (**C**$_*$ *Jean*) *aime*)]-[N:*Marie*]	(**>B**)	**Compilation**
4	[S:((**B** (**C**$_*$ *Jean*) *aime*) *Marie*)]	(**>**)	

A l'étape 2, au vu du résultat intermédiaire [S/(S\N) : (**C**$_*$ *Jean*)], nous pouvons comprendre grâce au combinateur **C**$_*$ que la règle déclenchée pour arriver à ce résultat est une règle de changement de type. Avec le type N de *Jean* et le type S/(S\N) de (**C**$_*$ *Jean*), nous déduisons que la règle déclenchée est (**>T**) (voir cette règle).

A l'étape 3, le résultat intermédiaire est [S/N : (**B** (**C**$_*$ *Jean*) *aime*)]. Le combinateur **B** nous renseigne sur la règle déclenchée pour obtenir ce résultat. C'est une règle de composition fonctionnelle bien sûr. D'autre part, avec les types syntaxiques respectifs de (**C**$_*$ *Jean*), *aime*, et (**B** (**C**$_*$ *Jean*) *aime*), nous déduisons que la règle déclenchée est (**>B**) (voir cette règle).

Garder la trace des opérations est utile, car ceci nous permet de reconsidérer notre analyse dans les cas de décomposition en particulier et ainsi concevoir des retours arrière intelligents.

Par ailleurs, l'utilisation des combinateurs dans une expression applicative autorise un calcul qui nous permet de retrouver le type syntaxique de cette expression, et ce à partir des types syntaxiques de ses composants élémentaires.

Le calcul en question est fondé sur les résultats suivants :

i - Pour la règle (**>B**) (respectivement (**>Bx**)), **B** f g s'applique à un opérande de type syntaxique Z placé à droite (respectivement à gauche) (comme pour g), afin de former une structure de type syntaxique X (comme pour f).
Pour la règle (**<B**) (respectivement (**<Bx**)), **B** g f s'applique à un opérande de type syntaxique Z placé à gauche (respectivement à droite) (comme pour f), afin de former une structure de type syntaxique X (comme pour g).

ii - La règle (**>S**) (respectivement (**>Sx**)) construit un opérateur complexe **S** f g, qui comme g, s'applique à un opérande de type syntaxique Z placé à droite (respectivement à gauche) de façon à produire comme f, un résultat de type syntaxique X.
La règle (**<S**) (respectivement (**<Sx**)) construit un opérateur complexe **S** g f qui comme f s'applique à un opérande de type syntaxique Z placé à gauche (respectivement à droite) de façon à produire comme g, un résultat de type syntaxique X.

iii - Dans l'expression applicative (f g), l'ordre applicatif associe à f le statut d'opérateur et à g le statut d'opérande. g est ainsi opérande de f.
Pour connaître le type syntaxique de (f g) il suffit de connaître celui de f. En effet, nous savons que si le type syntaxique de f est X/Y ou X\Y alors Y est le type syntaxique de g et X est le type syntaxique de (f g).

Prenons un exemple pour illustrer cela :

Calculons le type syntaxique de la structure applicative :

((**B** (**C**$_*$ *Jean*) *mange*) (*la soupe*))

Au départ nous avons, enregistré dans une table, les types de (**C**$_*$ *Jean*)[21], *mange*, *la* et *soupe* qui sont respectivement S/(S\N), (S\N)/N, N/N et N.

La première structure que nous mettons en évidence est (f g), où f est représenté par (**B** (**C**$_*$ *Jean*) *mange*) et g est représenté par (*la soupe*). Ainsi pour connaître le type syntaxique de ((**B** (**C**$_*$ *Jean*) *mange*) (*la soupe*)), il suffit de connaître celui de (**B** (**C**$_*$ *Jean*) *mange*) et de récupérer le type du résultat de l'application de celui-ci à son opérande (voir le résultat iii).

Nous ne connaissons pas le type de (**B** (**C**$_*$ *Jean*) *mange*). Il faut le calculer. Dans la structure (**B** f g), f est représenté par (**C**$_*$ *Jean*) et g par *mange*. Nous avons les types de (**C**$_*$ *Jean*) et *mange* dans un tableau. Nous pouvons donc calculer le type de (**B** (**C**$_*$ *Jean*) *mange*). (**B** f g) prend pour argument, l'argument de g et l'application de (**B** f g) à son argument a pour type syntaxique celui de l'application de f à son argument (voir le résultat i). Ainsi, (**B** (**C**$_*$ *Jean*) *mange*) prendra pour argument, l'argument de *mange* et l'application de (**B** (**C**$_*$ *Jean*) *mange*) à son argument aura pour type syntaxique celui de l'application de (**C**$_*$ *Jean*) à son argument. Nous déduisons ainsi que le type syntaxique de (**B** (**C**$_*$ *Jean*) *mange*) est S/N.

(**B** (**C**$_*$ *Jean*) *mange*) étant opérateur dont l'opérande est (*la soupe*), en prenant le type résultant de l'application de (**B** (**C**$_*$ *Jean*) *mange*) à son argument nous récupérons le type syntaxique de ((**B** (**C**$_*$ *Jean*) *mange*) (*la soupe*)), autrement dit S.

5. OUVERTURE VERS D'AUTRES LANGUES

Nous avons vu dans la section 4 comment utiliser des règles d'un calcul catégoriel combinatoire dans le traitement d'une langue naturelle : le français.

Comment envisager le traitement d'autres langues ? Quelles modifications ce traitement induirait ?

Nous supposons que le traitement d'une langue comme l'anglais n'induirait pas beaucoup de modifications à notre système, ce dernier étant principalement conçu pour le traitement des langues SVO. Mais, nous pensons que des modifications plus importantes, par rapport au système actuel, s'imposent pour traiter une langue VSO comme l'arabe (l'arabe se lit de droite à gauche, pour simplifier la lecture de ce papier, nous supposerons qu'il se lit de gauche à droite, ceci n'engendre pas d'incohérence dans notre explication).

---

[21] Nous ne savons pas encore calculer le type syntaxique de (C$_*$ u1) à partir du type syntaxique de u1. C'est la raison pour laquelle nous enregistrons ces structures dans une table.

Pour être plus précis, les règles de changement de type et de composition à elles seules ne sont pas suffisantes pour rendre du traitement d'une phrase aussi simple que :

akala	*Zayd*	*alkhoubza*	
(mangea)	(Zayd)	(le pain)	*Zayd mangea le pain*

En effet, le verbe *akala* (manger) est un opérateur qui prend deux opérandes : le sujet et l'objet. Nous lui donnons le type syntaxique (S/N)/N. Le sujet s'intercale entre le verbe et l'objet. Or, l'objet est considéré comme étant le premier argument de l'opérateur verbe. Nous ne pouvons évidemment pas accepter cette analyse :

akala	*Zayd*	*alkhoubza*
[(S/N)/N : *akala*]	[N : *Zayd*]	[N : *alkhoubza*]
[S/N : (*akala Zayd*)]		
[S : ((*akala Zayd*) *alkhoubza*)]		

L'analyse que nous estimons adéquate, permute les positions des opérandes du verbe. Cette permutation s'obtient avec l'introduction dans la chaîne syntagmatique du combinateur **C**. Nous aurons ainsi à la fin de l'analyse syntaxique l'expression applicative suivante :

(((**C** *akala*) *Zayd*) *alkhoubza*)

La réduction du combinateur **C** dans cette expression donne l'expression applicative sans combinateur suivante :

(*akala alkhoubza*) *Zayd*)

C'est cette interprétation sémantique fonctionnelle que nous acceptons.

Les règles de permutation[22] qui introduisent le combinateur **C** sont :

[(X\Y)/Z : f]		[(X/Y)/Z : f]
----------**>C**	;	----------**>Cx**
[(X/Z)\Y : (**C** f)]		[(X/Z)/Y : (**C** f)]
[(X/Y)\Z : f]		[(X\Y)\Z : f]
----------**<C**	;	----------**<Cx**
[(X\Z)/Y : (**C** f)]		[(X\Z)\Y : (**C** f)]

Avec ces règles la bonne analyse de la phrase *akala Zayd alkhoubza* est la suivante :

---

[22] Les règles (**>C**) et (**<C**) sont des théorèmes du calcul de Lambek. (**>Cx**) et (**<Cx**) ne le sont pas (du moins, nous n'arrivons pas à prouver qu'elles le sont).

phénotype

1	[(S/N)/N : *akala*]-[N : *Zayd*]-[N : *alkhoubza*]	
2	[(S/N)/N : (C *akala*)]-[N : *Zayd*]-[N : *alkhoubza*]	(>Cx)
3	[S/N : ((C *akala*)*Zayd*)]-[N : *alkhoubza*]	(>)
4	[S : (((C *akala*) *Zayd*) *alkhoubza*)]	(>)

génotype

5	[S : (((C *akala*) *Zayd*) *alkhoubza*)]	
6	[S : ((*akala alkhoubza*) *Zayd*)]	(C)

Les règles de permutation comme, nous venons de le montrer sont fort utiles dans le traitement de langues non SVO. Toutefois, nous ne savons pas encore quelles seront les incidences sur notre système sachant que ((**B** (**C**$_*$ x) f) y) ≡ (((**C** f) x) y). Ne risque-t-il pas de se présenter des incohérences dans un système avec des règles de changement de type et des règles de permutation ? Faudrait-il imposer des restrictions sur l'utilisation des règles de changement de type ? Quelles sont les critères linguistiques qui seraient susceptibles de motiver l'utilisation des règles de permutation ? Autant de questions qui restent ouvertes.

6. CONCLUSION

Avec ce travail, nous sommes arrivés à la conviction que les Grammaires Catégorielles associées à un formalisme applicatif peuvent rendre compte correctement des langues naturelles. Même si, beaucoup de choses restent à faire pour établir un système prêt à répondre à des besoins industriels, nous pensons que ce que nous présentons est assez solide pour être considéré comme un modèle de traitement des langues naturelles relativement fiable.
Ce modèle, d'ailleurs, présente des qualités que nous jugeons importantes. En premier lieu, nous trouvons qu'il est élégant dans le sens où il permet un suivi de traitement très aisé. Deuxièmement, l'utilisation combinée des types syntaxiques et des combinateurs permet de véhiculer un maximum d'informations avec un minimum de ressources (variables). Nous avons pu apprécier cela dans la section 5. Cette propriété, même si nous sommes en face uniquement d'une étude théorique, a l'avantage de réduire (voire éliminer) des traitements que nous dirons superflus, comme celui fondé sur l'unification pour retrouver les interprétations sémantiques dans le modèle de Steedman et Pareschi.

Par ailleurs, le modèle de la GCCA prend la forme d'une compilation. On ne fait pas un traitement sémantique parallèle à un traitement syntaxique. La construction de l'interprétation sémantique fonctionnelle se fait dans la continuité du calcul syntaxique avec le passage du phénotype au génotype.

Comme nous l'avons signalé précédemment, ce travail s'inscrit dans un cadre de travail beaucoup plus général. D'autres traitements dans le génotype suivent celui que nous avons présenté. Nous pensons en particulier au traitement des paraphrases, de la passivation et de la reflexivation, *etc.*. Ainsi, nous pouvons conseiller au lecteur de consulter (Desclés, 90) pour comprendre comment peuvent se faire ces genres de traitements.

ANNEXE

La notion de combinateur fut introduite pour la première fois par Schönfinkel (1924), elle est reprise par Curry et Feys en 1958. La logique combinatoire a été développée pour analyser logiquement les paradoxes (Russell) et la notion de substitution. Elle est aussi en rapport avec le λ-calcul de Church (1941). Ces deux systèmes d'ailleurs sont les moyens utilisés actuellement par les informaticiens pour analyser les propriétés sémantiques des langages de programmation de haut niveau.

Les combinateurs sont des opérateurs abstraits qui construisent, des opérateurs plus complexes à partir d'opérateurs plus élémentaires. L'action d'un combinateur sur un argument est définie par une règle spécifique appelée *β-réduction* [23]. Cette règle établit une relation indépendante de la signification des arguments entre une expression avec un combinateur et une expression sans combinateur équivalente à la première (d'un certain point de vue).

Donnons les règles d'élimination de quelques combinateurs[24] élémentaires :

B f g x
----------élimination-**B**
f (g x)
S f g x
----------élimination-**S**
f x (g x)
Φ f g h x
----------élimination-**Φ**
f (g x) (h x)
$\mathbf{C}_*$ x f
----------élimination-$\mathbf{C}_*$
f x
C X Y Z
----------élimination-C
X Z Y

---

[23] Dans la terminologie de Curry.

[24] Outre les combinateurs élémentaires que nous venons de donner, il existe des combinateurs complexes construits à partir des combinateurs élémentaires. Nous avons par exemple : **B B** $\mathbf{C}_*$

L'action de ces combinateurs est déterminée par l'application enchaînée des combinateurs élémentaires à partir du combinateur élémentaire le plus à gauche.

1	**B B** $\mathbf{C}_*$ x y z	
2	**B** ($\mathbf{C}_*$ x) y z	élimination-**B**
3	($\mathbf{C}_*$ x) (y z)	élimination-**B**
4	y z x	élimination-$\mathbf{C}_*$

BIBLIOGRAPHIE

ABEILLE, A., 1993, *Les nouvelles syntaxes : Grammaires d'unification et analyse du français,* Paris, Armand Colin

AJDUKIEWICZ, K., 1935, "Die syntaktishe Konnexität", *Studia philosophica*, vol. 1, 1-27.

ADES, A., STEEDMAN, M., 1982, "On the order of words", *Linguistics and Philosophy* , 4, 517-558.

BACH, E., 1988, "Categorial Grammars as theories of language" dans Oehrle *et alii*, 1988, 17-34.

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

BARRY, G., PICKERING, M., 1990, "Dependancy and constituency in categorial grammar" dans Lecomte 1992, *L'ordre des mots dans les grammaires catégorielles,* 38-57.

BISKRI, I., DESCLES, J.P., 1995, "La Grammaire Catégorielle Combinatoire Applicative", *T.A. Information.*

BISKRI, I., DESCLES, J.P., 1995, "Applicative and Combinatory Catégorial Grammar (from syntax to functional semantics)", Acte du *Colloque RANLP, Bulgarie 1995.*

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

BUSKOWSKI, W., 1988, "Generative Power of Categorial Grammar", dans Oehrle et alii, 1988, 69-94.

CURRY, B. H., FEYS, R., 1958, *Combinatory logic* , Vol. I, North-Holland.

DESCLES, J.P., GUENTCHEVA, Z., SHAUMYAN, S.K., 1985, *Theoritical aspects of passivization in the framework of Applicative Grammar*, John Benjamins.

DESCLES, J.P., GUENTCHEVA, Z., SHAUMYAN, S.K., 1986, "A theoritical analysis of reflexivization in the framework of Applicative Grammar", *Linguisticae Investigationes*, 2.

DESCLES, J. P., 1990 *Langages applicatifs, langues naturelles et cognition*, Paris, Hermès.

DESCLES, J. P., SEGOND, F., 1990, "Topicalization : categorial analysis and applicative grammar" dans Lecomte 1992, *L'ordre des mots dans les grammaires catégorielles,* 13-37.

DOWTY, D., 1988, "Type-Raising, Functional Composition, and Non-Constituent Conjunction" dans Oehrle et alii, 1988.

GARDIES, J.L., 1975, *Esquisse d'une grammaire pure*, Paris, Vrin.

GUENTCHEVA, Z., 1976, "Présentation critique du modèle applicationnel de S.K. Shaumyan", *Documents de linguistique quantitative* , n°30, Dunod.

HADDOCK, E. KLEIN, et G. MORILL, 1987,*Working papers in cognitive science volume I : Categorial Grammar, Unification Grammar and Parsing,* Edinburgh University Press.

HEPPLE, M., 1991 "Efficient incremental processing with Categorial Grammar" *Actes du colloque ACL* 1991

HINDLEY, J. R., SELDIN, J. P., 1986, *Introduction to Combinators and Lambda-Calculus,* Cambridge, Cambridge Univ. Press.

HUSSERL, E., *Logische Untersuchungen*, Halle, Max Niemeyer, 1913.

JOUIS, C., 1993, *Contributions à la conceptualisation et à la modélisation des connaissances à partir d'une analyse linguistique de textes. Réalisation d'un prototype : le système SEEK,* Thèse de doctorat, EHESS, Paris 1993.

LAMBEK, J., 1958, "The Mathematics of Sentence Structure", *American Mathematical Monthly*, 65, 154-165.

LAMBEK, J., 1961, "On the calculus syntactic types" *Proceeding of symposia in Applied Mathematics,* vol. XII, America Mathematical Society, Providence, Rhode Island, 166-178.

LECOMTE, A., 1992, *Word order in Categorial Grammar / L'ordre des mots dans les Grammaires Catégorielles*, Adosa.

LECOMTE, A., 1994, *Modèles Logiques en Théorie Linguistique : éléments pour une théorie informationnelle du langage* ; Synthèse de travaux présentés en vue de l'habilitation à diriger des recherches.

LESNIEWSKI, S.T., 1989, *Sur les fondements de la mathématique. Fragments (Discusions préalables, méréologie, ontologie).* Traduit du polonais par Georges Kalinowski, Paris, Hermès.
MONTAGUE, R., 1974, *Formal philosophy : selected papers of Richard Montague*, R. Thomason (ed), New Haven, Yale University Press.
MOORTGAT, M., 1988, *Categorial Investigation, Logical and Linguistic Aspects of the Lambek calculus*, Foris Publications.
OEHRLE, R.T., BACH, E., WHEELER, D., 1988, *Categorial Grammars and Natural Languages Structures*, D. Reidel.
PARESCHI, R., STEEDMAN, M., 1987, "A lazy Way to chart parse with categorial grammars" *Acte du colloque ACL,* Stanford, 1987.
PARTEE, B.H., MEULEN, A., WALL, R.E., 1993, *Mathematical Methods in Linguistics*, Kluwer Academic Publishers.
SEGOND, F., 1990, *Grammaire catégorielle du français. Etude théorique et implantation. Le système GraCe (Grammaire catégorielle étendue)*, Nouveau doctorat, EHESS-Paris
SHAUMYAN, S. K., 1977, *Applicational Grammar as a Semantic Theory of Natural Language*, Edimburgh Univ. Press.
SHAUMYAN, S. K., 1987, *A Semiotic Theory of Natural Language*, Bloomington, Indiana Univ. Press.
STEEDMAN, M., 1988, "Combinators and Grammars", dans Oehrle *et alii*, 1988, 207-263.
STEEDMAN, M., 1989, "Constituency and coordination in a combinatory grammar" dans M. BALTIN and T. KROCH, *Alternative conceptions of phrase structure*, University of Chicago press, 201-231.
STEEDMAN, M., 1989, *Work in progress: Combinators and grammars in natural language understanding*, Summer Institute of linguistic, Tucson University.
SZABOLCSI, A., 1987, "On combinatory categorial grammar", *Acte du Symposium on logic and languages,* Debrecen, Akademiai Kiado, Budapest, 151-162.