Structures mathématiques du langage

Le calcul de Lambek dans ses présentations
“déduction naturelle ” et “calcul des séquents”

Résumé

Dans ce chapitre, nous traitons du calcul de Lambek pur, un système de règles et d'axiomes qui permet de représenter les dérivations syntaxiques comme des preuves au sein d'une logique qui s'apparente à la fois à la logique intuitionniste (et qui, de ce fait, peut être présenté aussi bien sous la forme de la Déduction Naturelle que sous celle du calcul des séquents) et à la logique linéaire (à cause de sa sensibilité aux ressources, aussi bien dans leur quantité que dans leur disposition gauche-droite). L'intérêt majeur de ce calcul réside dans la manière dont il permet le calcul de représentations sémantiques, en se basant uniquement sur l'isomorphisme de Curry-Howard, c'est-à-dire la correspondance que l'on peut établir entre logique et calcul, les types correspondant à des formules et les preuves à des λ-termes.

Table des matières

1 Grammaires catégorielles et calcul logique

1.1 Grammaires de Lambek

Nous partirons des deux règles suivantes, dites d'application fonctionnelle, déjà vues dans le chapitre sur la grammaire de Montague :

FA avant : $A/B \;\; B \quad \rightarrow \quad A$

FA arrière : $B \;\; B\backslash A \quad \rightarrow \quad A$

Mais il s'avère vite que ces deux règles sont insuffisantes si on veut rendre compte d'une manière qui ne soit pas *ad hoc* de certaines constructions avec déplacement (au sens de Chomsky) comme dans l'exemple suivant :

Exemple 1 Mary read the book that she bought

Nous pouvons admettre une assignation très intuitive de catégories syntaxiques aux mots de cette phrase :

read, bought	:	$(np\backslash s)/np$
Mary	:	np
she	:	$s/(np\backslash s)$
book	:	n
the	:	np/n
that	:	$(n\backslash n)/(s/np)$

Commentaire : ainsi, *that* donne un modifieur de nom s'il trouve sur sa droite une phrase à laquelle il manque un syntagme nominal. Le pronom *she* reçoit une catégorie complexe (et pas une simple comme np) qui prend en compte le fait qu'il peut apparaître seulement comme sujet et non comme objet. Sous cette forme, on a : *she* donne une phrase quand il est complété par une phrase à laquelle il manque un sujet.
Nous obtenons une liste de catégories syntaxiques :

$$np \;\; (np\backslash s)/np \;\; np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; s/(np\backslash s) \;\; (np\backslash s)/np \tag{1}$$

et cette suite ne peut pas être réduite car il n'y a pas de np à la périphérie droite qui pourrait satisfaire l'attente du verbe. Le problème sera résolu si on ajoute d'autres règles, comme :

Composition Fonctionnelle

FC avant :	$A/B \;\; B/C$	$\rightarrow$	A/C
FC arrière :	$C\backslash B \;\; B\backslash A$	$\rightarrow$	$C\backslash A$

De fait, ceci conduit à la succession suivante de pas de réduction :

1. *FC avant* :
$$np \quad (np\backslash s)/np \quad np/n \quad n \quad (n\backslash n)/(s/np) \quad \frac{s/(np\backslash s) \;\; (np\backslash s)/np}{s/np}$$
2. *FA avant* :
$$np \quad (np\backslash s)/np \quad np/n \quad n \quad \frac{(n\backslash n)/(s/np) \;\; s/np}{n\backslash n}$$
3. *FA arriere*
$$np \quad (np\backslash s)/np \quad np/n \quad \frac{n \;\; n\backslash n}{n}$$
4. *FA avant* :
$$np \quad (np\backslash s)/np \quad \frac{np/n \;\; n}{np}$$
5. *FA avant* :
$$np \quad \frac{(np\backslash s)/np \;\; np}{np\backslash s}$$

$$\frac{A \quad A\backslash B}{B} \qquad \frac{B/(A\backslash B) \quad A\backslash B}{B}$$

$$\frac{B/A \quad A}{B} \qquad \frac{B/A \quad (B/A)\backslash B}{B}$$

FIG. 1 – Type lifting

6. *FA arriere* :

$$\frac{np \;\; np\backslash s}{s}$$

Ce qui est surprenant ici est que la règle de Composition Fonctionnelle "déplace un *trou*" (le np manquant) depuis la périphérie droite et vers la gauche jusqu'à ce qu'il soit absorbé par une catégorie qui, précisément valide ce trou (ici le complémenteur *that*). Un tel argument manquant correspond à la notion chomskyenne de trace, mais au lieu d'avoir un mouvement de np qui laisserait une trace derrière lui, c'est le trou, autrement dit la trace qui se déplace jusqu'à l'opérateur qui le valide[1]. Pour cette raison, un tel cadre peut être vu comme "sans déplacement" et sans catégories vides.
Ce cadre prévoit aussi d'autres règles, qui deviendront utilisables, comme l'élévation de type. Nous avons :

Elévation de Type

TR avant : $A \rightarrow B/(A\backslash B)$
TR arrière : $A \rightarrow (B/A)\backslash B$

Ces règles expriment le fait que tout objet linguistique de type A peut être vu comme un objet attendant sur sa droite un objet attendant sur sa gauche un objet de type A, ou symmétriquement, comme un objet attendant sur sa gauche un objet attendant sur sa droite un objet de type A. Cela est mis en évidence sur la figure 1. Notons ici que, comme J. Lambek le montre dans son article de 1958 (voir section suivante), ces types élevés peuvent être exemplifiés au moyen des pronoms. Notre première intuition concernant les pronoms anglais *he* (ou *she* etc.) et *him* (ou *her* etc.) consiste à leur assigner le type n, comme c'est la cas pour les noms propres (ou np quand la différenciation sera faite entre ces deux types, ce qui n'est pas le cas dans les premiers stades de la grammaire de Lambek), mais il n'y aurait alors aucune différence entre :

he likes Mary
et **Mary likes he*

ni entre :

Mary likes him
et **him likes Mary*

[1]Cette analyse des dépendances entre *remplisseurs* (*fillers*) et *trous* (*gaps*) par-delà une distance arbitraire à l'intérieur d'une phrase a été adoptée dans d'autres cadres, comme HPSG (*Head Driven Phrase Structure Grammar*), voir [**?**].

puisque, dans les deux cas, les deux expressions se réduisent à s. Quel type donc devrions-nous donner aux pronoms *he* et *him* ? Considérons la phrase : *he sleeps*, l'autre assignation de type qui permettrait sa réduction à s consiste à donner à *he* le type $s/(n\backslash s)$. En effet, dans ce cas, nous aurions :

he	*sleeps*
$s/(n\backslash s)$	$n\backslash s$
s	

D'une façon similaire, nous pouvons donner à *him* le type $(s/n)\backslash s$. On note alors que, dans ce cas, on obtient la réduction suivante :

$$\begin{aligned} Mary\ likes\ him & \qquad (2)\\ n\ \ ((n\backslash s)/n\ \ (s/n)\backslash s) & \qquad (3)\\ (n\ \ n\backslash(s/n))\ \ (s/n)\backslash s & \qquad (\text{Assoc.})\\ s/n\ \ (s/n)\backslash s & \qquad (\text{FA backward})\\ s & \qquad (\text{FA backward}) \end{aligned}$$

à condition de permettre l'utilisation d'une règle selon laquelle les parenthèses peuvent être modifiées autour des types dans des expressions comme : $n\backslash(s/n)$, autrement dit la règle :

$$(Assoc) \quad X\backslash(Y/Z) \leftrightarrow (X\backslash Y)/Z$$

Si, cependant, nous voulons une dérivation pour :

Exemple 2 he likes him

alors il apparaît que nous restons bloqués sans l'utilisation des règles FC. Avec elles, nous obtenons l'arbre de réduction suivant :

$$\dfrac{s/(n\backslash s) \qquad \dfrac{n\backslash(s/n) \quad (s/n)\backslash s}{n\backslash s}\ FC\ backward}{s}\ FA\ forward$$

Notons finalement que, grâce à TR, nous avons :

$$n \rightarrow s/(n\backslash s) \qquad n \rightarrow (s/n)\backslash s$$

donc tout nom peut être élevé au type d'un pronom, mais :

$$s/(n\backslash s) \not\rightarrow n \qquad (s/n)\backslash s \not\rightarrow n$$

autrement dit ; un pronom ne peut pas être "abaissé" au type d'un nom, de sorte qu'en assignant à *he* (resp. *him*), le type élevé $s/(n\backslash s)$ (resp. $(s/n)\backslash s$) *dans le lexique*, nous n'obtiendrons pas d'abaisement de ce type à n, tandis qu'en assignant le type n à un nom quelconque, nous serons toujours capables de de l'élever en cas de besoin.
Mais le temps est venu de mettre un peu d'ordre dans cet ensemble de règles.

1.2 Le calcul de Lambek (pur)

1.2.1 La mathématique des phrases

L'article de Lambek de 1958 (*The Mathematics of Sentence Structure*[Lambek(1958)]) est la première tentative visant à mettre en évidence, sous la notion de grammaire catégorielle, un véritable système logique. Le calcul proposé par Lambek est "logiquement complet" en deux sens :

– il l'est au sens où il est possible de lui doner une interprétation sémantique vis-à-vis de laquelle il est complet (autrement dit, le système permet l'énumération de toutes les "vérités" de l'interprétation sémantique)
– et il l'est aussi au sens où il fournit un système complet de règles : en ajouter d'autres le ferait dégénérer, c'est-à-dire accepter beaucoup trop de phrases en comparaison de ce qui est désiré (cf. infra).

Le but que se donnait Lambek était réellement :

> d'obtenir une règle effective (ou un algorithme) permettant de distinguer les phrases des non-phrases, qui s'appliquerait non seulement aux langages formels qui intéressent les logiciens et les mathématiciens, mais aussi aux langues naturelles comme l'Anglais, ou au moins à des fragments significatifs de telles langues.

Lambek se base sur l'idée de Bar-Hillel que les catégories grammaticales contiennent des "/" aussi bien que des "\". De plus, il rejette l'idée d'assignation rigide (un seul type par mot ou expression) mais c'est pour faire en sorte que, dans la mesure du possible, les types variés que peut revêtir une expression puissent se déduire à partir d'un type plus élémentaire.

1.2.2 Un système catégorique

Lambek montre que tous les schémas de règles vus plus haut peuvent être déduits, comme de simples théorèmes, à l'intérieur d'un système formel composé d'axiomes et de quelques règles, où x, y, z dénotent des types syntaxiques, où nous admettons que si une expression A est de type x, une expression B de type y, alors l'expression AB (obtenue par simple concaténation) est de type xy, ou x•y (ce qui suppose que le calcul ne contient pas seulement / et \, mais aussi un troisième opérateur, •, qui est un *produit non commutatif* entre les types). De plus, nous écrivons "$x \rightarrow y$"[2] pour dire qu'une expression de type x a aussi le type y.

(a)	x → x		
(b)	(x•y)•z → x•(y•z)	(b')	x•(y•z) → (x•y)•z
(c)	if x•y → z then x → z/y	(c')	if x•y → z then y → x\ z
(d)	if x → z/y then x•y → z	(d')	if y → x\ z then x•y → z
(e)	if x → y and y → z then x → z		

Les formules (a), (b) et (b') sont évidemment des axiomes : (a) exprime l'identité, (b) et (b') l'*associativité*. (c), (c'), (d), (d') et (e) donnent des règles : ils montrent comment de nouvelles flèches peuvent être ajoutées au système. La règle (e) exprime la *transitivité* de la flèche. Les lecteurs déjà informés de la Théorie des Catégories, au sens mathématique du terme, reconnaissent les axiomes nécessaires pour obtenir une catégorie : *identité* et *transitivité*. Dans ce cas, les formules seront interprétées comme des *objets* et les flèches comme des *morphismes*. Cette sorte de catégorie est appelée un *système déductif*.
Il est intéressant de faire, dans le cadre de ce système, la démonstration de certaines des règles que nous avons postulées plus haut.

Elévation de type :

$$\dfrac{\dfrac{\dfrac{}{z/y \rightarrow z/y}\,(a)}{(z/y)\ y \rightarrow z}\,(d)}{y \rightarrow (z/y)\backslash z}\,(c')$$

[2]Cette flèche est la flèche de réduction, qui sera remplacée plus tard par le symbole"⊢".

Composition Fonctionnelle :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y/x \to y/x}(a)}{(y/x)\bullet x \to y}(d) \quad \dfrac{}{y \to (z/y)\backslash z}Th1}{(y/x)\bullet x \to (z/y)\backslash z}(e)}{(z/y)\bullet((y/x)\bullet x) \to z}(d')}{((z/y)\bullet(y/x))\bullet x \to z}(b')}{(z/y)\bullet(y/x) \to z/x}(c)$$

Exercices :
Démontrer de même :

1. $z/y \to (z/x)/(y/x)$ (règle de division)
2. $(x/y)/z \leftrightarrow x/(z \bullet y)$ (règle de curryfication)
3. si $x \to x'$ et $y \to y'$ alors $x \bullet y \to x' \bullet y'$
4. si $x \to x'$ et $y \to y'$ alors $x/y' \to x'/y$

1.2.3 Le calcul de Lambek vu comme calcul des séquents

Existe-t-il un système de décision pour le calcul de Lambek ? Lambek rejoint ici Gentzen. Un calcul des séquents pour le calcul de Lambek se distingue du cas général par le fait que d'abord évidemment, les valuations par "vrai" ou "faux" n'y ont pas de sens, ensuite parce que, a priori, aucune des règles structurelles n'est présente. Dans un calcul comme **L**, aussitôt qu'une formule est utilisée comme prémisse dans une déduction, elle disparaît, on peut dire que dans une déduction de type Lambek, chaque prémisse est utilisée **une et une seule fois** : ceci est le caractère **linéaire** du calcul de Lambek. Mais ce calcul est plus que "linéaire", il est aussi non commutatif (absence de la règle de permutation), et comme toutes les déductions qui nous intéressent (pour l'instant) sont des réductions à une catégorie (la catégorie S par exemple) et que, de ce fait, elles prennent la forme d'arbres, nous ne nous intéressons qu'aux séquents qui n'ont qu'une seule formule en partie droite. Ceci est le caractère **intuitionniste** du calcul de Lambek. En réunissant toutes ces considérations, nous pouvons arriver à l'ensemble suivant de règles.

$$\frac{}{A \vdash A}\,[Axiom]$$

$$\frac{\Theta \vdash B \qquad \Gamma, A, \Delta \vdash C}{\Gamma, A/B, \Theta, \Delta \vdash C}\,[/L] \qquad\qquad \frac{\Theta \vdash B \qquad \Gamma, A, \Delta \vdash C}{\Gamma, \Theta, B\backslash A, \Delta \vdash C}\,[\backslash L]$$

$$\frac{\Gamma, B \vdash A}{\Gamma \vdash A/B}\,[/R] \qquad\qquad \frac{B, \Gamma \vdash A}{\Gamma \vdash B\backslash A}\,[\backslash R]$$

$$\frac{\Gamma \vdash A \qquad \Delta, A, \Delta' \vdash B}{\Delta, \Gamma, \Delta \vdash B}\,[Cut]$$

Il est très simple de prouver dans ce système la validité d'un séquent correspondant à une phrase, en construisant sa preuve *sans coupure*. Considérons par exemple la phrase :

Exemple 3 the cat that Paul feeds hits the bottle

avec les assignations de types suivantes :

the	np/n
cat	n
that	$(n\backslash n)/(s/np)$
Paul	np
feeds	$(np\backslash s)/np$
hits	$(np\backslash s)/np$
bottle	n

nous obtenons la déduction suivante, où δ représente la sous-preuve de droite.

$$\dfrac{\dfrac{\dfrac{}{np \vdash np}[Ax] \quad \dfrac{\dfrac{}{np \vdash np}[Ax] \quad \dfrac{}{s \vdash s}[Ax]}{np, np\backslash s \vdash s}[\backslash L]}{np, (np\backslash s)/np, np \vdash s}[/L]}{np, (np\backslash s)/np \vdash s/np}[/R] \quad \dfrac{\delta}{np/n, n, n\backslash n, (np\backslash s)/np, np/n, n \vdash s}}{np/n, n, (n\backslash n)/(s/np), np, (np\backslash s)/np, (np\backslash s)/np, np/n, n \vdash s}[/L]$$

où δ consiste en :

$$\dfrac{\dfrac{}{n \vdash n}[Ax] \quad \dfrac{\dfrac{}{n \vdash n}[Ax] \quad \dfrac{\dfrac{}{np \vdash np}[Ax] \quad \dfrac{\dfrac{}{np \vdash np}[Ax] \quad \dfrac{}{s \vdash s}[Ax]}{np, np\backslash s \vdash s}[\backslash L]}{np, (np\backslash s)/np, np \vdash s}[/L]}{np, (np\backslash s)/np, np/n, n \vdash s}[/L]}{np/n, n, (np\backslash s)/np, np/n, n \vdash s}[/L]}{np/n, n, n\backslash n, (np\backslash s)/np, np/n, n \vdash s}[\backslash L]$$

1.2.4 Presentation Déduction Naturelle

Avant d'aller plus au fond dans l'étude du calcul des séquents, nous pouvons évidemment noter que le calcul de Lambek peut aussi s'exprimer dans le cadre de la déduction naturelle. Les règles pour le calcul de Lambek (restreint à $/$ et $\backslash$, sans $\bullet$) peuvent s'exprimer en déduction naturelle comme :

$$\dfrac{A/B \qquad B}{A}[e\ /] \qquad\qquad \dfrac{B \qquad B\backslash A}{A}[e\ \backslash]$$

Commentaire : avec deux prémisses A/B et B (resp. B et $B\backslash A$) dans cet ordre, on peut déduire A (c'est exactement la règle d'application fonctionnelle, FA)

$$\dfrac{\Gamma \qquad \begin{array}{c}[B]^i \\ \vdots \\ A\end{array}}{A/B}[i\ /]^i \qquad\qquad \dfrac{\begin{array}{c}[B]^i \\ \vdots \\ A\end{array} \qquad \Gamma}{B\backslash A}[i\ \backslash]^i$$

Commentaire : si, d'une suite de types Γ à laquelle on ajoute sur la droite (resp. sur la gauche) une catégorie B, on déduit A, alors de Γ toute seule, on peut déduire A/B (resp. $B\backslash A$). L'indice i est utilisé pour indiquer quelle hypothèse est déchargée pendant l'utilisation de la règle - i.

1.2.5 Exemples

Examinons quelques exemples.

Exemple 4 Mary likes a Japonese writer

cette phrase donne lieu à la suite suivante de types syntaxiques :

Mary : np
likes : $(np\backslash s)/np$
a : $((s/np)\backslash s)/n$
Japonese : n/n
writer : n

Evidemment l'analyse de *a japonese writer* se fait facilement au moyen d'une application répétée des règles FA, ou, en termes plus logiques, des règles d'élimination :

$$\frac{a : ((s/np)\backslash s)/n \qquad \dfrac{japonese : n/n \quad writer : n}{japonese\ writer : n}\,[e\ /]}{a\ japonese\ writer : (s/np)\backslash s}\,[e\ /]$$

A ce stade, il n'y a pas de connexion directe entre le type verbal ($(np\backslash s)/np$) et le type d'un syntagme nominal, qui est le type d'ordre élevé $(s/np)\backslash s$. Nous savons intuitivement que les choses vont s'arranger en vertu des règles d'élévation de type selon lesquelles le np attendu par le verbe peut aussi être vu comme étant du type $(s/np)\backslash s$, mais au lieu d'utiliser ici cette connaissance externe, nous pouvons choisir de rester à l'intérieur du calcul et de résoudre le problème *en faisant usage d'hypothèses*. Introduisons donc une hypothèse np juste à droite du verbe (avec l'indice 1) :

Mary : np likes : $(np\backslash s)/np$ $[np]^1$ a japonese writer : $(s/np)\backslash s$

Ceci conduit aux sous-preuves successives suivantes :

$$Mary : np \qquad \frac{likes : (np\backslash s)/np \quad [np]^1}{likes\ \epsilon : np\backslash s} \qquad a\ japonese\ writer : (s/np)\backslash s$$

$$\frac{Mary : np \qquad \dfrac{likes : (np\backslash s)/np \quad [np]^1}{likes\ \epsilon : np\backslash s}}{Mary\ likes\ \epsilon : s} \qquad a\ japonese\ writer : (s/np)\backslash s$$

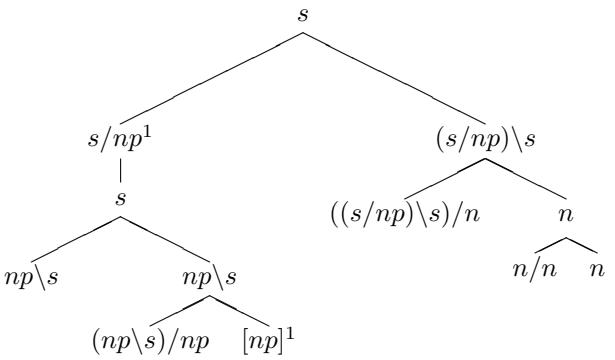
A cette étape, l'hypothèse np peut être déchargée, et nous obtenons :

$$\frac{\dfrac{Mary : np \qquad \dfrac{likes : (np\backslash s)/np \quad [np]^1}{likes\ \epsilon : np\backslash s}}{Mary\ likes\ \epsilon : s}}{Mary\ likes : s/np}\,[i\ /]^1 \qquad a\ japonese\ writer : (s/np)\backslash s$$

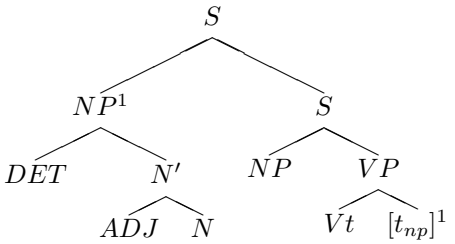
avec maintenant la partie gauche qui peut être appliquée à la partie droite, par la règle d'élimination ($[e\ \backslash]$)

$$\frac{\dfrac{\dfrac{Mary : np \qquad \dfrac{likes : (np\backslash s)/np \quad [np]^1}{likes\ \epsilon : np\backslash s}}{Mary\ likes\ \epsilon : s}}{Mary\ likes : s/np}\,[i\ /]^1 \qquad a\ japonese\ writer : (s/np)\backslash s}{Mary\ likes\ a\ japonese\ writer : s}\,[e\ \backslash]$$

Informellement, nous avons utilisé la concaténation entre chaînes pour représenter quelle expression chaque type obtenu recouvre durant la déduction. ϵ (la chaîne vide) est utilisé pour marquer phonologiquement une hypothèse.
Il est intéressant de représenter cet arbre de preuve à la façon d'un arbre de dérivation syntaxique en le renversant, ce qui donne :



Il est alors éclairant de le comparer avec un arbre syntaxique qui serait davantage dans la tradition générative, incluant une transformation de montée du quantifieur :



Ils sont évidemment très similaires : dans le premier arbre, l'hypothèse a le rôle joué par la trace dans le second. Mais dans le premier arbre, le syntagme nominal quantifié ne se déplace pas, à proprement parler : c'est le trou destiné à le recevoir qui se déplace, depuis la position de complément du verbe jusqu'au niveau de la phrase dans son ensemble, avant d'être absorbé par l'attente du syntagme nominal.

Exemple 5 Which book do you think that Mary read ?

Cette question donne lieu à la suite suivante de types :

which : $(s/(s/np))/n$	book : n	do : s/s	you : $s/(np\backslash s)$
think : $(np\backslash s)/np$	that : np/s	Mary : np	read : $(np\backslash s)/np$

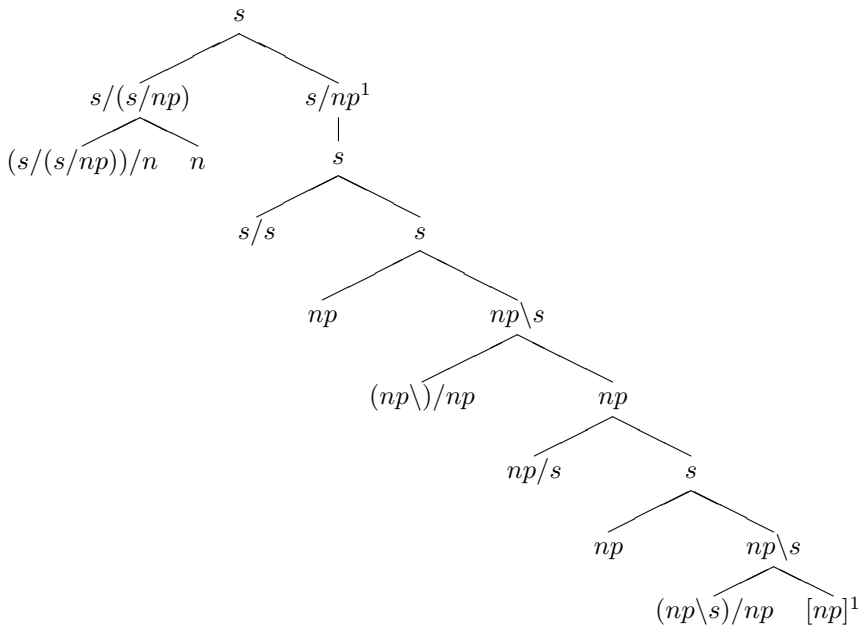
that Mary read est analysé comme suit :

$$\dfrac{\dfrac{that : np/s \qquad \dfrac{Mary : np \qquad \dfrac{read : (np\backslash s)/np \quad (hyp) : [np]^1}{read\ \epsilon : np\backslash s}}{Mary\ read\ \epsilon : s}}{that\ Mary\ read\ \epsilon : np}}{that\ Mary\ read : np/np}\ [i\ /]^1$$

de manière similaire, nous obtenons :

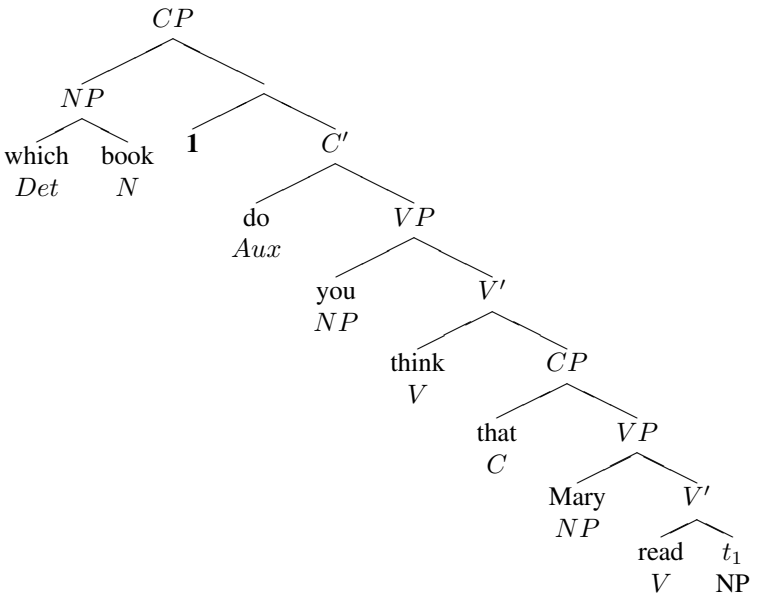
$$do\ you\ think\ that\ Mary\ read : s/np$$

et par plusieurs pas d'élimination de $/$, nous obtenons le typage s de la phrase entière. L'arbre de preuve inversé[3] est donné par :

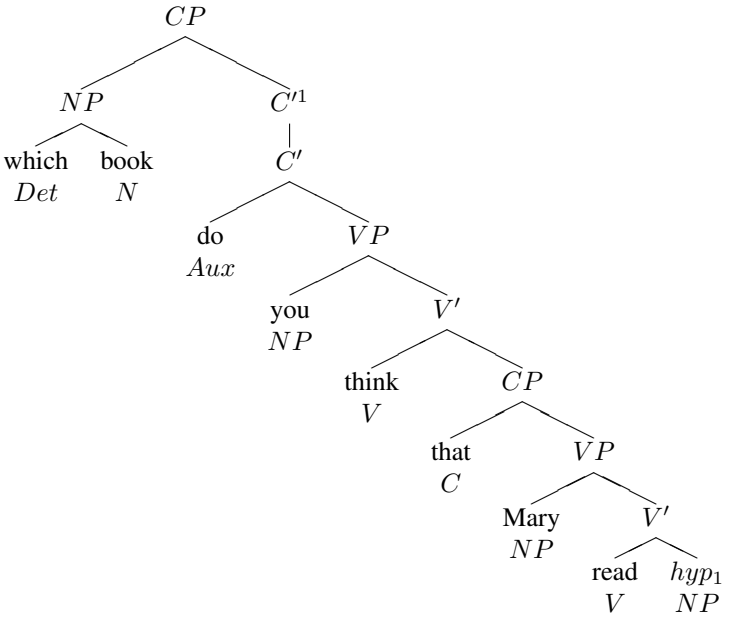


[3] "Inversé" signifie ici : dont les règles sont présentées dans l'ordre inverse, la conclusion étant en haut et les prémisses en bas.

Cet arbre de preuve peut être comparé avec la dérivation, pour la même phrase, dans une grammaire à la Heim et Kratzer :



et avec notre intuition selon laquelle le mécanisme de déplacement laissant une trace derrière lui est similaire au raisonnement avec hypothèse. Au lieu d'introduire un lieur (*binder*) **1**, il semble plus approprié d'avoir une branche unaire exprimant le pas où l'hypothèse est déchargée :



1.2.6 Sémantique compositionnelle

Dans la sémantique de Montague, comme dans celle de Heim et Kratzer, l'abstraction λ doit nécessairement être utilisée : ceci est le cas dans les règles S_3 et S_{14} de la grammaire de Montague, ainsi que dans

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

$$\frac{\Gamma \qquad \begin{array}{c}[B]^i\\ \vdots\\ A\end{array}}{A/B}\,[i\ /]^i \qquad\qquad \frac{\begin{array}{c}[B]^i\\ \vdots\\ A\end{array} \qquad \Gamma}{B\backslash A}\,[i\ \backslash]^i$$

FIG. 2 – Lambek rules in Natural Deduction

l'opération d'abstraction de prédicat de Heim et Kratzer. Ici, en appliquant la correspondance de Curry – Howard, nous obtenons directement 2.

Supposons qu'un terme f soit associé à A/B ou à $B\backslash A$ dans les règles d'élimination, et qu'un terme τ soit associé à B, alors, à A sera associée l'*application* de f à τ.

Supposons d'autre part qu'une *variable* soit associée à chaque hypothèse, par exemple x à $[B]^i$ et que la conclusion de la preuve dans les règles d'introduction, soit associée au terme f, alors en déchargeant l'hypothèse, nous obtenons le λ-term $\lambda x.f$. Ainsi, les règles données plus haut peuvent-elles être complétées de la manière suivante :

$$\frac{A/B : f \qquad B : \tau}{A : f(\tau)}\,[e\ /] \qquad\qquad \frac{B : \tau \qquad B\backslash A : f}{A : f(\tau)}\,[e\ \backslash]$$

$$\frac{\Gamma \qquad \begin{array}{c}[B : x]^i\\ \vdots\\ A : f\end{array}}{A/B : \lambda x.f}\,[i\ /]^i \qquad\qquad \frac{\begin{array}{c}[B : x]^i\\ \vdots\\ A : f\end{array} \qquad \Gamma}{B\backslash A : \lambda x.f}\,[i\ \backslash]^i$$

La figure 3 montre cette correspondance dans le format du calcul des séquents, où $\gamma[x \leftarrow \mu]$ est la substitution de μ à x partout où x apparaît dans l'expression γ, et où la variable x n'a pas déjà été utilisée dans les termes associés aux autres types apparaissant dans le séquent.

Comme on peut le noter, la règle de coupure a une interprétation fonctionnelle en tant que *substitution* dans un terme β.

En utilisant la *correspondance de Curry – Howard* entre les preuves et les λ-termes, nous pouvons calculer les représentations sémantiques des deux phrases précédentes (ici sans les chaînes étiquetant les types dans leur partie "phonologique"). (voir les figures 4 et 5).

1.3 Algorithme de décision et élimination des coupures

Nous allons ici traiter de points qui pourraient être étudiés dans un cadre plus général. Le calcul des séquents fournit un algorithme de décision en logique propositionnelle intuitionniste, et le problème de l'élimination des coupures est résolu en logique des prédicats classique (Gentzen, 1934) et même dans des systèmes bien plus évolués comme le système F de J-Y. Girard (1975). Néanmoins, voir ces sujets dans le cadre du calcul de Lambek est agréable car plus simple, sans toutefois que l'on masque les traits essentiels.

$$\frac{}{A : x \vdash A : x}\,[Axiom]$$

$$\frac{\Theta \vdash B : \alpha \qquad \Gamma, A : x, \Delta \vdash C : \gamma}{\Gamma, A/B : f, \Theta, \Delta \vdash C : \gamma[x \leftarrow f(\alpha)]}\,[/L]$$

$$\frac{\Theta \vdash B : \alpha \qquad \Gamma, A : x, \Delta \vdash C : \gamma}{\Gamma, \Theta, B\backslash A : f, \Delta \vdash C : \gamma[x \leftarrow f(\alpha)]}\,[\backslash L]$$

$$\frac{\Gamma, B : x \vdash A : f}{\Gamma \vdash A/B : \lambda x.f}\,[/R] \qquad\qquad \frac{B : x, \Gamma \vdash A : f}{\Gamma \vdash B\backslash A : \lambda x.f}\,[\backslash R]$$

$$\frac{\Gamma \vdash A : \alpha \qquad \Delta, A : x, \Delta' \vdash B : \beta}{\Delta, \Gamma, \Delta \vdash B : \beta[x \leftarrow \alpha]}\,[Cut]$$

FIG. 3 – Règles du calcul de Lambek en calcul des séquents

$$\dfrac{np : mary \qquad \dfrac{(np\backslash s)/np : \lambda u \lambda v.like(v,u) \quad [x : np]^1}{np\backslash s : \lambda v.like(v,x)}}{\dfrac{\dfrac{s : like(mary,x)}{s/np : \lambda x.like(mary,x)}\,[i\ /]^1 \qquad (s/np)\backslash s : \lambda P.\exists z(JW(z) \wedge P(z))}{s : \exists z(JW(z) \wedge like(mary,z))}\,[e\ \backslash]}$$

FIG. 4 – Mary likes a japonese writer

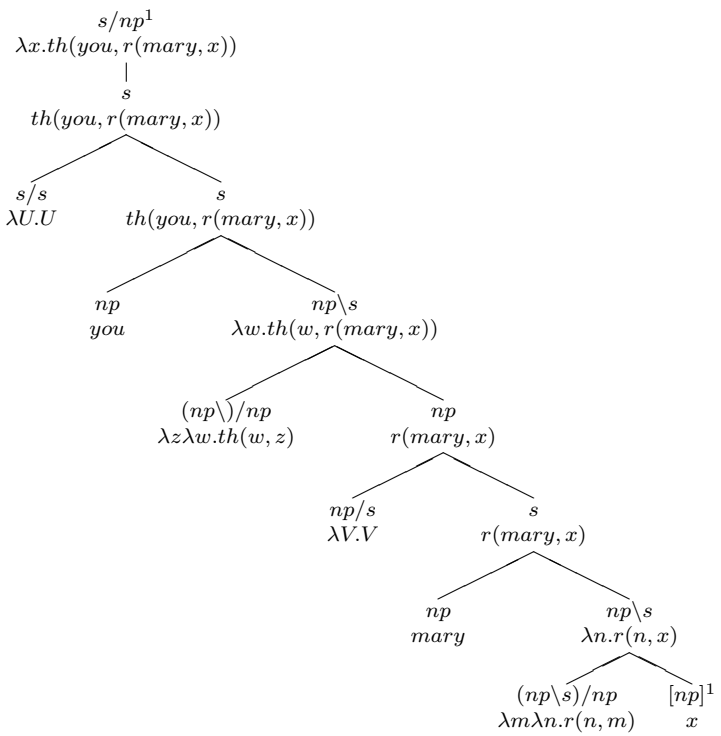


FIG. 5 – you think that Mary read ? (here in the inverted proof tree format)

1.3.1 Propriété de la sous-formule

Comme nous l'avons vu plus haut (Exemple 3), il est très facile de trouver une déduction sans coupure pour une phrase telle que *the cat that Paul feeds hits the bottle*, et d'une manière générale, étant donnée une suite de catégories syntaxiques $c_1, c_2, ..., c_n$ nous arriverons toujours à déterminer en un temps fini s'il existe une réduction de cette suite à une catégorie syntaxique c : autrement dit, le calcul des séquents sans coupure est *décidable*. Cela tient aux faits suivants :

- nous pouvons toujours commencer par utiliser toutes les règles droite que l'on peut jusqu'à obtenir le problème de réduire une suite $c'_1, c'_2, ..., c'_n$ à un type atomique c'. En effet, nous cherchons quel est le connecteur principal de la catégorie c et nous appliquons la règle d'introduction à droite de ce connecteur à notre séquent jusqu'à ce qu'il n'y ait plus de connecteur dans le type en partie droite,
- nous sélectionnons au hasard une catégorie complexe en partie gauche et nous tentons d'utiliser la règle d'introduction à gauche de son connecteur principal, découpant son contexte de manière arbitraire et décomposant ainsi cette formule et appliquant à ses sous-formules la même méthode jusqu'à ce qu'on arrive à des constituants atomiques. Si nous rencontrons un blocage, nous revenons en arrière et essayons un autre découpage du contexte et ainsi de suite.
- comme les listes et les formules sont finies, les découpages possibles de suites de formules sont également en nombre fini on arrive donc toujours au bout d'un temps plus ou moins long, mais fini, soit à une situation où toutes les branches se terminent par un axiome (succes), soit à une situation où plus aucune possibilité n'est à explorer et où pourtant on demeure bloqué (échec).

La réussite de cet algorithme tient à la propriété dite "de la sous-formule" :
- **Propriété de la sous-formule** : *toutes les règles de* **L***, sauf la règle de coupure, sont telles que la ou les prémisse(s) contien(nen)t et ne contien(nen)t que des sous-formules des formules du séquent conclusion.*
Autrement dit, il n'y a pas de formule à deviner quand on part du séquent à prouver et qu'on monte par application des règles (lues de bas en haut) vers des axiomes (seuls des découpages éventuels du contexte sont à deviner, mais on a vu qu'il n'y en avait qu'un nombre fini)[4]. La règle de coupure est une exception notable : la formule de coupure A est ici à deviner ! Les démonstrations utilisant la règle de coupure demandent donc plus d'habileté. Mais obtient-on plus de choses avec la règle de coupure que sans ? Ici, apparaît un résultat surprenant, démontré par Gentzen en 1934, et qui porte le nom de **Hauptsatz** :
- **Théorème d'élimination des coupures** : *le système de Gentzen (pour la logique des prédicats du premier ordre classique) permet de démontrer les mêmes théorèmes en utilisant la règle de coupure et sans utiliser la règle de coupure !.*
Nous allons esquisser la preuve de ce théorème à propos du calcul de Lambek dans le paragraphe suivant.

1.3.2 Elimination des coupures

Si nous appelons **L** l'ensemble des règles du calcul des séquents de Lambek, le théorème se ramène à dire que **L** et **L** $-Cut$ ont la même capacité déductive. Ceci est bien sûr étonnant car la règle de coupure n'est pourtant pas une *conséquence* des autres règles au sens où on pourrait la démontrer au moyen d'une déduction utilisant les autres règles : il n'y a pas de déduction *directe* de Cut à partir de $/L$, $\backslash L$, $/R$, $\backslash R$ etc. La démonstration de redondance de la règle de coupure se fait en montrant comment elle peut être systématiquement éliminée d'une preuve quelconque, en remplaçant pour chaque élimination d'une occurrence de Cut la preuve donnée par une preuve dont les Cut jouent sur des formules de plus en plus simples, jusqu'à ce qu'ils ne jouent que sur des atomes, auquel cas on peut alors tous les enlever.
Nous suivrons Lambek dans son article de 1958 pour cette démonstration.

Définition 1 *- Le degré $d(x)$ d'une formule x est son nombre d'occurrences de connecteurs $/$ ou $\backslash$*
- Le degré d'une suite de formules est la somme des degrés des formules

[4]Malheureusement, ceci ne s'applique pas au calcul des prédicats puisqu'en ce cas, pour les formules avec quantificateur, les sous-formules peuvent être en nombre infini.

- Le degré d'un cut :

$$\frac{T \vdash x \qquad U, x, V \vdash y}{U, T, V \vdash y}$$

est la somme des degrés de T, de U, de V, de x et de y

Démonstration cas par cas :

1. cas où $T \vdash x$ est un axiome. On a :

$$\frac{x \vdash x \qquad U, x, V \vdash y}{U, x, V \vdash y}$$

   autrement dit le Cut ne sert à rien et il peut être éliminé, on se retrouve simplement avec le séquent $U, x, V \vdash y$.

2. cas où $U, x, V \vdash y$ est un axiome. Alors U et V sont vides et $x = y$. on a :

$$\frac{T \vdash x \qquad x \vdash x}{T \vdash x}$$

   comme précédemment, le Cut ne sert à rien. On se retrouve avec $T \vdash x$.

3. cas où la dernière règle appliquée pour arriver à $T \vdash x$ était une des règles de **L** sauf Cut et $Axiome$, mais n'introduisait pas le connecteur principal de x. Alors cette règle était une règle gauche et l'une de ses deux prémisses[5] était forcément de la forme $T' \vdash x$ où T' a un degré plus faible que T (puisqu'une règle a fait disparaître un connecteur en passant du bas vers le haut). On peut considérer l'instance suivante de Cut

$$\frac{T' \vdash x \qquad U, x, V \vdash y}{U, T', V \vdash y}$$

   elle est de degré inférieure à la précédente. De plus, la règle qui a permis de passer de $T' \vdash x$ à $T \vdash x$ peut maintenant sans difficulté nous faire passer de $U, T', V \vdash y$ à $U, T', V \vdash y$! De sorte qu'on obtient :

$$\dfrac{\Sigma \qquad \dfrac{T' \vdash x \qquad U, x, V \vdash y}{U, T', V \vdash y}\, Cut}{U, T, V \vdash y}\, r$$

   à la place de :

$$\dfrac{\dfrac{\Sigma \qquad T' \vdash x}{T \vdash x}\, r \qquad\qquad U, x, V \vdash y}{U, T, V \vdash y}\, Cut$$

   (où Σ est l'autre prémisse nécessaire pour employer la règle r) ce qui montre qu'on est passé d'une preuve avec Cut à une preuve avec un Cut de degré moins élevé.

4. cas où le dernier pas dans la preuve de $U, x, V \vdash y$ utilise une des règles de **L** sauf Cut et $Axiome$ mais sans introduire le connecteur principal de x. Alors, $U, x, V \vdash y$ est déduit à partir d'une ou de deux prémisses dont l'une a la forme $U', x, V' \vdash y'$. Comme dans le cas précédent, la suite

[5] il y a deux prémisses si la règle est l'introduction à gauche de / ou de \. Dans le calcul de Lambek avec produit,l'introduction du produit à gauche n'aurait qu'une seule prémisse.

U', x, V', y' a un degré plus faible que la suite U, x, V, y et donc le Cut suivant est de degré plus faible :

$$\frac{T \vdash x \qquad U', x, V' \vdash y'}{U', T, V' \vdash y'}$$

et comme dans le cas précédent, la même règle r faisant passer de $U', x, V' \vdash y'$ à $U, x, V \vdash y$ peut nous faire passer de $U', T, V' \vdash y'$ à $U, T, V \vdash y$, autrement dit on peut encore permuter la règle r et le Cut.

5. cas où le dernier pas des preuves des deux prémisses concerne, dans les deux cas, la formule x. Prenons l'exemple de $x = x'/x''$ (le cas de $x''\backslash x'$ serait symétrique, et on laisse au lecteur le soin de traiter le cas du produit si on tient au produit dans **L**). Alors, on a d'un côté application de la règle $/R$ et de l'autre de la règle $\backslash R$. La preuve se présente donc comme suit :

$$\frac{\dfrac{T, x'' \vdash x'}{T \vdash x'/x''}\,/R \quad \dfrac{V' \vdash x'' \qquad U, x', V'' \vdash y}{U, x'/x'', V', V'' \vdash y}\,/L}{U, T, V', V'' \vdash y}\,Cut$$

Elle peut être entièrement réaménagée de la manière suivante :

$$\frac{V' \vdash x'' \qquad \dfrac{T, x'' \vdash x' \qquad U, x', V'' \vdash y}{U, T, x'', V'' \vdash y}\,Cut}{U, T, V', V'' \vdash y}\,Cut$$

où les deux Cut qui apparaissent ont un degré plus faible que le Cut de la première preuve.

1.3.3 L'interprétation computationnelle de l'élimination des coupures

Limitons-nous ici à la logique intuitionniste implicationnelle positive (fragment de la logique intuitionniste, qui se distingue donc du calcul de Lambek par le fait qu'il admet toutes les règles structurelles alors que ce dernier n'en admet aucune, cela évite d'avoir une distinction entre $/$ et $\backslash$, confondus ici en la simple implication intuitionniste $\Rightarrow$). Imaginons dans cette logique, une portion de preuve contenant une occurrence de la règle de coupure [6] :

$$\frac{\dfrac{\Delta, y : A \vdash f : B}{\Delta \vdash \lambda y.f : A \Rightarrow B}\,[\Rightarrow D] \quad \dfrac{\Gamma \vdash \alpha : A \qquad \Gamma, x : B \vdash \gamma : C}{\Gamma, u : A \Rightarrow B \vdash [(u\ \alpha)/x]\gamma : C}\,[\Rightarrow G]}{\Delta, \Gamma \vdash [(\lambda y.f\ \alpha)/x]\gamma : C}\,coupure$$

(Remarque : $\lambda y.f$ se substitue à u dans le terme $[(u\ \alpha)/x]\gamma$, d'où le résultat).
L'algorithme remplace cette coupure par deux autres instances de cette même règle, mais de "degré" inférieur, on obtient le fragment de preuve suivant :

$$\frac{\dfrac{\Delta, y : A \vdash f : B \qquad \Gamma \vdash \alpha : A}{\Delta, \Gamma \vdash [\alpha/y]f : B} \qquad \Gamma, x : B \vdash \gamma : C}{\Delta, \Gamma \vdash [[[\alpha/y]/f]/x]\gamma : C}\,coupure$$

Ce faisant, le λ-terme $(\lambda y.f\ \ \alpha)$, substitué à x dans γ, s'est transformé en $[\alpha/y]f$, qui est justement le résultat de la β-réduction du terme $(\lambda y.f\ \ \alpha)$. Une étape dans l'algorithme d'élimination de la règle

[6] Ceci représente le cas, dans la démonstration du théorème d'élimination de la coupure, où deux règles symétriques se rencontrent, l'une d'introduction à gauche l'autre à droite du même connecteur.

de coupure s'est donc traduite en une étape de calcul dans la β-*réduction* d'un λ-terme. Ceci permet de compléter la correspondance de Curry-Howard : non seulement, les propositions sont des types, les preuves sont des λ-termes, les règles d'introduction et d'élimination de $\Rightarrow$ respectivement les opérations d'abstraction et d'application, mais en plus, la normalisation des preuves est la β-réduction.

Références

[Church(1941)] Church, A. (1941). *The calculi of lambda-conversion, Annals of mathematical studies* , 6, pp. ii–77.

[Curry(1961)] Curry, H. (1961). *Some logical aspects of grammatical structure,* in R. Jakobson (ed.), *Structure of Language and its Mathematical Aspects* (Providence), pp. 56–68.

[Curry and Feys(1958)] Curry, H. and Feys, R. (1958). *Combinatory Logic vol. 1* (North-Holland, Amsterdam).

[Gamut(1991)] Gamut, L.-T.-F. (1991). *Logic, Language and Meaning, vol. I and II* (The University of Chicago Press).

[Heim and Kratzer(1998)] Heim, I. and Kratzer, A. (1998). *Semantics in Generative Grammar* (Blackwell, Malden, Mass).

[Lambek(1958)] Lambek, J. (1958). *The Mathematics of Sentence Structure, American Mathematical Monthly* **65**, pp. 154–170.

[Lambek(1961)] Lambek, J. (1961). *On the Calculus of Syntactic Types, Structure of Language and its Applications* .

[Lambek(1988)] Lambek, J. (1988). *Categorial and categorical grammars,* in E. Bach, R. Oehrle and D.Wheeler (eds.), *Categorial Grammars and Natural Language Structures* (D. Reidel), pp. 297–317.

[Moortgat(1988)] Moortgat, M. (1988). *Categorial Investigations, Logical and Linguistic Aspects of the Lambek Calculus* (Foris, Dordrecht).

[Moortgat(1997)] Moortgat, M. (1997). *Categorial Type Logics,* in [van Benthem and ter Meulen (1997)], chap. 2, pp. 93–178.

[Morrill(1994)] Morrill, G. (1994). *Type Logical Grammar, Categorial Logic of Signs* (Kluwer, Dordrecht).

[van Benthem and ter Meulen (1997)] van Benthem, J. and ter Meulen, A. (eds.) (1997). *Handbook of Logic and Language* (Elsevier).