LES GRAMMAIRES DE LAMBEK RIGIDES NE SONT PAS APPRENABLES À PARTIR DES CHAÎNES.

Annie Foret*, Yannick Le Nir*

IRISA Rennes, FRANCE
foret@irisa.fr et ylenir@irisa.fr

Résumé

Cet article traite de l'apprentissage des grammaires catégorielles selon le modèle de Gold (Gol67). Récemment, des algorithmes d'apprentissages ont été proposés dans ce modèle pour quelques classes particulières de grammaires catégorielles (Kan98). Nous montrons que contrairement aux grammaires catégorielles classiques, les grammaires de Lambek rigides ou k-valuées ne sont pas apprenables à partir des chaînes. Ce résultat est valable pour plusieurs variantes du calcul de Lambek. Notre preuve revient à construire un point limite pour chaque classe. Ces résultats permettent de mieux cerner les directions dans lesquelles on peut envisager de nouveaux algorithmes d'apprentissage.

1 INTRODUCTION

Les grammaires catégorielles ont été étudiées dans le domaine du traitement automatique du langage naturel. Ce modèle de la syntaxe des langues entretient des rapports étroits avec la sémantique (à la Montague). Les grammaires catégorielles classiques (dites AB) sont apparues dans (BH53) ; nous nous intéressons ici aux grammaires catégorielles de Lambek (Lam58) qui en sont un raffinement.

Les grammaires de Lambek fonctionnent avec des règles (fixées) de nature logique sur les types ; analyser une phrase revient à démontrer un théorème sur les types associés aux mots de la phrase ; le système de déduction logique correspondant peut aussi être vu comme une variante de la logique linéaire de Girard (Gir95).

Les grammaires catégorielles sont lexicalisées : elles associent des types (ou catégories) à chaque mot d'un lexique; lorsqu'elles associent au plus k types à chaque symbole du lexique, on dira qu'elles sont k-valuées.

De telles grammaires k-valuées ont attiré l'attention lors de récents travaux sur l'apprentissage (Kan98) (Nic99). Acquérir une grammaire revient dans ce cadre à acquérir les types associés aux mots. De tels algorithmes d'acquisition d'une

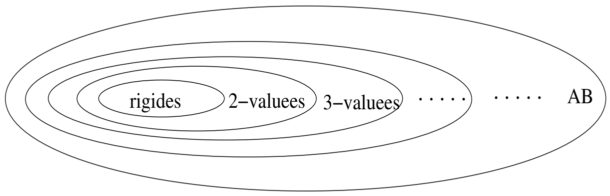


FIG. 1 – *Inclusion des classes de grammaires catégorielles classiques k-valuées*

grammaire formelle sont utiles pour les applications courantes du traitement automatique des langues telles que correction orthographique, aide à la traduction, interrogation en langage naturel.

Ces résultats soulèvent de nouvelles questions, l'extension des travaux de Kanazawa à d'autres classes de grammaires catégorielles ou leur amélioration est ainsi devenu depuis peu un thème de recherche très actif. Le lien avec la sémantique est aussi utilisé dans cette perspective, par exemple dans (DSTT01).

Dans ce contexte, il est très important d'acquérir une bonne compréhension des propriétés des différentes classes de grammaires en question. Rappelons tout d'abord que les grammaires catégorielles de Lambek sont faiblement équivalentes aux grammaires hors contexte, c'est à dire qu'elles génèrent les mêmes langages. Ce résultat, obtenu par Pentus (Pen93), est une première justification à la recherche de sous classes potentiellement apprenables. En effet, alors que les grammaires catégorielles classiques ne sont pas apprenables puisque faiblement équivalentes aux grammaires hors contextes (YBHS60), les sous classes k-valuées de ces grammaires sont apprenables d'après les résultats de Kanazawa.

Cependant, alors que les classes des grammaires catégorielles k-valuées classiques sont bien identifiées (cf figure 1) , les classes des langages reconnaissables par des grammaires de Lambek k-valuées ne sont pas encore bien caractérisées. Il apparaît pertinent d'examiner ces classes de grammaires dont certaines sont apprenables à partir de structures d'après (BR01). Dans ce papier, nous traitons le problème suivant, est-ce que la classe des grammaires de Lambek rigides est apprenable à partir de chaînes.

L'apprentissage (au sens de Gold (Gol67)) dans notre contexte est un procédé symbolique pouvant être présenté comme suit. Soit $\mathcal{G}$ une classe de grammaires que nous voulons apprendre à partir d'exemples positifs. Le but est de définir un algorithme qui appliqué à un ensemble fini de phrases, retourne une grammaire de la classe qui a généré les exemples; l'algorithme devant converger.

Formellement, soit $\mathcal{L}(G)$ le langage associé à la grammaire G, et V un alphabet donné, un algorithme d'apprentissage est une fonction ϕ d'ensembles finis de mots dans V^* vers $\mathcal{G}$ telle que pour $G \in \mathcal{G}$ avec $\mathcal{L}(G) =< e_i >_{i\in\mathbb{N}}$ il existe une grammaire $G' \in \mathcal{G}$ et il existe $n_0 \in N$ tel que: $\forall n > n_0$ $\phi(\{e_1, \ldots ,e_n\}) = G' \in \mathcal{G}$ avec $\mathcal{L}(G') = \mathcal{L}(G)$.

L'utilisation du modèle de Gold a pendant plusieurs années semblé impossible pour les classes non triviales de langages, en conséquence du résultat initial de non apprenabilité de (Gol67). Le regain d'intérêt pour l'apprentissage à la limite à partir d'exemples positifs est dû aux travaux de Dana Angluin (Ang80), qui détermina une première classe non triviale de grammaires apprenables. Ce résultat, en plus de relancer l'intérêt pour ce genre d'apprentissage, permit également d'identifier le premier exemple de classe intéressante de grammaires transversales à la hiérarchie de Chomsky. L'abandon définitif du pessimisme pour ce modèle d'apprentissage fut effectif avec le résultat de Shinohara (Shi90) qui montra que les grammaires contextuelles k-rigides sont également apprenables. Les résultats d'apprentissage de Kanazawa (Kan98) viennent compléter la liste des grammaires non triviales apprenables avec les grammaires catégorielles k-valuées classiques dont l'apprentissage se révèle particulièrement intéressant de par leur complète lexicalisation: les règles sont déjà connues, seuls les types associés aux mots doivent être dérivés à partir des exemples.

Le papier est organisé comme suit. La section 2 présente les définitions nécessaires et les résultats connus. La section 3 donne ensuite la construction initiale et la preuve pour le calcul de Lambek avec séquence vide. Dans la section 4, nous étendons ce résultat au calcul de Lambek sans séquence vide avec produit. La section 5 conclut.

2 Définition et prérequis

2.1 Grammaires catégorielles

Dans cette partie, nous présentons quelques définitions de base sur les grammaires catégorielles. Le lecteur intéressé pourra également consulter (Cas88; Ret00; Bus97; Moo97) pour une introduction et de plus amples détails.

2.1.1 Exemples de typage

Dans ce qui suit, nous illustrons certains aspects du système de typage dans les grammaires catégorielles en utilisant des exemples linguistiques d'après (Lam58).

Types primitifs. Pour décrire des types syntaxiques associés aux mots (de phrases) nous commençons par introduire des types primitifs : S pour le type des phrases (bien construites) et N pour le type des noms y compris les noms propres tels que *Pierre* ou *Marie* ou encore des noms tels que *milk*,...

Des expressions composées peuvent aussi recevoir le type N, par exemple "pauvre Pierre" ou "fresh milk".

Types composés. Nous pouvons aussi introduire des types composés : en général une expression de type $x \ / \ y$ suivie par une expresssion de type y produit une expression de type x ; de façon semblable, une expression de type $y \setminus x$ précédée par une expression de type y produit une expression de type x.

Dans des expressions composées telles que "pauvre Pierre", on associe à l'adjectif "pauvre" un type composé $N \ / \ N$.

Un verbe intransitif tel que *travaille* dans “Pierre travaille” est associé au type $N \setminus S$. Un verbe transitif tel que *aime* est associé au type $(N \setminus S) \ / \ N$.

L’effet des regroupements. Dans certaines phrases, on peut regrouper les mots de différentes mainières : comme illustré par “Pierre (aime Marie)” et “(Pierre aime) Marie”.

Ainsi, certains mots comme les verbes transitifs peuvent être associés à plusieurs types : *aime* peut être associé à $(N \setminus S) \ / \ N$ ou encore à $N \setminus (S \ / \ N)$ dans les phrases ci-dessus.

Il se trouve que ces deux types sont équivalents dans le système L de typage proposé par Lambek (ils ne sont pas équivalents dans les grammaires AB).

Les pronoms et la montée de type. Dans des exemples (en anglais) tels que “he works”, or “he hates him” le pronom *he* [1] est associé au type $(S \ / \ (N \setminus S))$ et *him* est associé au type $((S \ / \ N) \setminus S)$.

Mais une expression de type N devrait aussi pouvoir être associée aux types de ces pronoms; ceci s’obtient par dérivation entre les types, ici par la règle dite de montée de type (“type-raising”) règle qui est valable dans les grammaires de Lambek (mais pas dans les grammaires AB).

Limitation des grammaires AB. Les points précédents illustrent certaines limitations des grammaires AB tandis que le calcul de Lambek permet d’assigner des types *flexibles*.

De ce point de vue, les grammaires AB ont d’autres limites concernant les dérivations qui paraissent souhaitables entre types; elles reposent sur une conception trop strictement concaténative ; il leur manque en particulier les règles de transitivité (telles que déduire $A \ / \ C$ de $A \ / \ B$ et $B \ / \ C$) qui sont valables dans les grammaires de Lambek.

2.1.2 DÉFINITIONS FORMELLES

Nous notons de façon générale X^* l’ensemble des mots (comprenant le mot vide) sur un alphabet X et X^+ l’ensemble des mots non vides sur l’alphabet X. Par la suite, Σ désigne un alphabet donné qui pourra représenter les mots d’une langue naturelle.

Types. Les *Types (ou catégories)* sont construits à partir de Pr (ensemble des *types primitifs*) et de trois connecteurs binaires $/$, $\setminus$ et $\bullet$ pour le produit. Tp représente l’ensemble des types : $Tp ::= Pr \mid Tp \setminus Tp \mid Tp \ / \ Tp \mid Tp \bullet Tp$.
Pr contient un type particulier, noté S, également apppelé *type principal* qui est associé aux phrases correctes du langage. D’un point de vue logique, les connecteurs $/$, $\setminus$ sont des formes d’implications (linéaires).

Grammaires catégorielles. Une *grammaire catégorielle* sur Σ est une relation finie G entre Σ et Tp. Si $< c,A > \in G$, on dit que G associe A à c, et on écrit $G : c \mapsto A$.

1. des pronoms tels que *he* ne remplacent pas toujours un nom dans “poor Peter hates milk”, *he* ne peut remplacer *Peter*, ni *milk*

Séquents. Dans le cadre des grammaires catégorielles, un séquent noté $\Gamma \vdash C$ est une paire (Γ,C) où Γ est une suite de types (potentiellement vide) et $C \in Tp$ est un type. Γ est appelé l'antécédent et C la conclusion du séquent.

Un séquent est dit valide si le type C est dérivable à partir de Γ. La notion de dérivabilité dépend du calcul logique considéré (le calcul de Lambek par exemple).

Nous commençons par présenter les règles de dérivation pour les grammaires AB.

Derivation $\vdash_{AB}$ sur les types sans $\bullet$. La relation $\vdash_{AB}$ est la plus petite relation $\vdash$ entre Tp^+ and Tp, telle que pour tout $\Gamma,\Delta \in Tp^+$ et pour tout $A,B \in Tp$:
$A \vdash A$
si $\Gamma \vdash A$ et $\Delta \vdash A \setminus B$ alors $\Gamma,\Delta \vdash B$ (“Backward application”)
si $\Gamma \vdash B \ / \ A$ et $\Delta \vdash A$ alors $\Gamma,\Delta \vdash B$ (“Forward application”)

Nous présentons maintenant une formulation du calcul de Lambek, noté L, incluant le produit, constitué de règles d'introduction à gauche et à droite d'un séquent. Pour le calcul de Lambek sans produit, on enlève simplement les règles pour $\bullet$.

Dérivation dans Lambek $\vdash_L$. La relation $\vdash_L$ est la plus petite relation $\vdash$ entre Tp^+ et Tp, telle que pour tout $\Gamma,\Gamma' \in Tp^+$, pour tout $\Delta,\Delta' \in Tp^*$ et pour tout $A,B \in Tp$:

$$A \vdash A \qquad \frac{A,\Gamma \vdash B}{\Gamma \vdash A \setminus B}\setminus r \qquad \frac{\Gamma,A \vdash B}{\Gamma \vdash B \ / \ A}/r \qquad \frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma,\Gamma' \vdash A \bullet B}\bullet r$$

$$\frac{\Gamma \vdash A \quad \Delta,B,\Delta' \vdash C}{\Delta,\Gamma,A \setminus B,\Delta' \vdash C}\setminus l \quad \frac{\Gamma \vdash A \quad \Delta,B,\Delta' \vdash C}{\Delta,B \ / \ A,\Gamma,\Delta' \vdash C}/l \quad \frac{\Delta,A,B,\Delta' \vdash C}{\Delta,A \bullet B,\Delta' \vdash C}\bullet l$$

Si l'on remplace Tp^+ par Tp^* dans $\Gamma \in Tp^+$ au sein de la définition ci-dessus, nous obtenons une autre version du calcul de Lambek, avec séquence vide, que l'on notera $L_\emptyset$ associé à la relation de dérivation $\vdash_{L_\emptyset}$.

Notes.
- Nous rappelons que la règle de coupure (Cut) est satisfaite par $\vdash_L$ et $\vdash_{L_\emptyset}$:
  si $\Gamma,A,\Gamma' \vdash C$ et $\Delta \vdash A$ alors $\Gamma,\Delta,\Gamma' \vdash C$ (Cut)
- Nous notons également que $\Gamma \vdash_L C$ implique $\Gamma \vdash_{L_\emptyset} C$
- Les règles pour AB restent valables dans $\vdash_L$ et $\vdash_{L_\emptyset}$.

Langage. Soit G une grammaire catégorielle sur Σ. G *génère* une chaîne $c_1 \ldots c_n \in \Sigma^+$ ssi il existe des types $A_1, \ldots ,A_n \in Tp$ tels que : $G \ : c_i \mapsto A_i \ (1 \leq i \leq n)$ et $A_1, \ldots ,A_n \vdash_L S$. Le langage de G, noté $\mathcal{L}_L(G)$ est l'ensemble des chaînes générées par G.

Nous définissons de la même manière $\mathcal{L}_{L_\emptyset}(G)$ et $\mathcal{L}_{AB}(G)$ en remplacant $\vdash_L$ par $\vdash_{L_\emptyset}$ et $\vdash_{AB}$.

Notation. Dans certaines sections, nous utiliserons $\vdash$ au lieu de $\vdash_L$ ou $\vdash_{L_\emptyset}$. De la même manière nous pourrons écrire simplement $\mathcal{L}(G)$.

Grammaires rigides et k-valuées. Les grammaires catégorielles associant au plus k types à chaque symbole de l'alphabet sont appelées grammaires k-valuées; les grammaires 1-valuées sont également appelées grammaires rigides.

Exemple 1
Soit $\Sigma_1 = \{Pierre, Marie, aime\}$ et $Pr = \{S, N\}$ pour le type des phrases et des noms respectivement. Soit $G_1 = \{Pierre \mapsto N, Marie \mapsto N, aime \mapsto N \setminus (S \,/\, N)\}$. On obtient $(Pierre\ aime\ Marie) \in \mathcal{L}_L(G_1)$ car $(N,\ N \setminus (S \,/\, N),\ N\ \vdash_L S)$.
G_1 est une grammaire rigide (ou 1-valuée).

Exemple 2
Nous illustrons maintenant les différences entre les grammaires de Lambek rigides et les grammaires AB rigides. Nous considérons pour cela un exemple en italien, issu de (Ret00), afin de travailler sur un lexique assez simple (pas de pronom).

Soit $\Sigma_2 = \{Paolo, Cosa, guarda, passare\}$ et $Pr = \{S, N, V\}$ pour le type des phrases, des groupes nominaux et des groupes verbaux respectivement.

Soit $G_2 = \{Paolo \mapsto N,\ Cosa \mapsto (S \,/\, S) \,/\, N,\ guarda \mapsto S \,/\, V,\ passare \mapsto V \,/\, N\}$.

On obtient $(guarda\ passare\ paolo) \in \mathcal{L}_L(G_2)$ et $(Cosa\ guarda\ passare) \in \mathcal{L}_L(G_2)$, (dont la traduction est "Il regarde passer Paolo" et "Que regarde-t-il passer?") d'après[2] la figure 2 ; G_2 est une grammaire rigide (ou 1-valuée).

Cependant, cette grammaire ne permet pas de dériver ces deux phrases dans les grammaires catégorielles AB rigides. En effet, la dérivation de $(Cosa\ guarda\ passare)$ utilise la transitivité, non valide dans les grammaires catégorielles rigides classiques. On a donc besoin de deux types pour $guarda$, ce qui nous donne par exemple la grammaire : $G'_2 = G_2 \cup \{guarda \mapsto (S \,/\, N) \,/\, (V \,/\, N)\}$
On obtient $(guarda\ passare\ paolo) \in \mathcal{L}_{AB}(G'_2)$ et $(Cosa\ guarda\ passare) \in \mathcal{L}_{AB}(G'_2)$ d'après la figure 2, mais à la différence de G_2 pour le cas de Lambek, ici G'_2 pour le cas AB est une grammaire 2-valuée, non rigide.

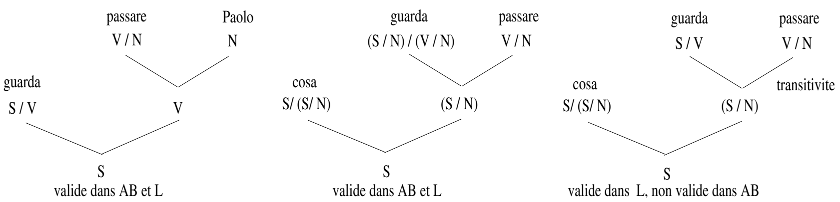


FIG. 2 – *Déductions avec et sans transitivité*

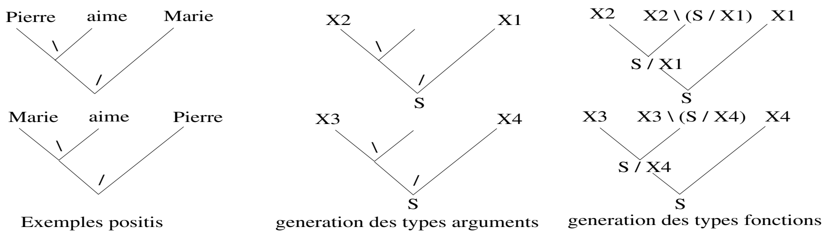
2.2 Inférence de grammaires catégorielles

Nous illustrons ici l'inférence grammaticale de grammaires catégorielles. Nous considérons le cadre le plus simple des grammaires AB rigides, avec des exemples

2. nous donnons ici une version graphique usuelle utilisant les règles de AB et la transitivité

positifs supposés structurés en arbres qui représentent la structure en constituants avec des étiquettes indiquant la forme des types (argument X ou fonction $X\ /\ Y$ ou $Y \setminus X$). W. Buszkowski et G. Penn ont montré pour cette classe qu'un algorithme d'unification sur les types permet de construire le lexique (la grammaire) le plus général engendrant les exemples positifs. Cette méthode a été utilisée et étendue dans (Kan98) pour certaines classes de grammaires AB. Nous donnons ci-dessous un exemple illustrant l'algorithme.

Exemple 3
A partir des exemples suivants,



nous unifions les types obtenus :

$Pierre$	X_2,X_3	$X_2 = X_3$	X_1
$aime$	$X_2 \setminus (S/X_1)$,$X_4 \setminus (S/X_3)$,	$X_2 = X_4$,$X_3 = X_1$	$X_1 \setminus (S/X_1)$
$Marie$	X_1,X_4	$X_1 = X_4$	X_1

2.3 Apprentissage et points limites

Nous rappelons maintenant quelques définitions utiles et quelques propriétés connues pour l'apprentissage.

Points limite. Une classe $\mathcal{CL}$ de langages a un point limite ssi il existe une chaîne infinie $< L_n >_{n \in N}$ de langages dans $\mathcal{CL}$ telle que : $L_0 \subsetneq L_1 \ldots \subsetneq ... \subsetneq L_n \subsetneq \ldots$ et $L = \bigcup_{n \in N} L_n$($L$ est un point limite de $\mathcal{CL}$).

Les points limite impliquent la non-apprenabilité. La propriété suivante est très importante pour notre propos. Si les langages des grammaires d'une classe $\mathcal{G}$ ont un point limite, alors la classe $\mathcal{G}$ n'est pas apprenable.[3]

2.4 Quelques modèles utiles

Pour simplifier les preuves dans les prochaines sections, nous utilisons des modèles pour L (ou $L_\emptyset$) dont nous allons rappeler la définition : *"powerset residuated"* semi-groupes (ou monoïdes selon le cas), un cas particulier des *"residuated"* semi-groupes (cf (Bus97) pour plus de détails).

3. Ceci implique que la classe à une élasticité infinie. Une classe $\mathcal{CL}$ de langages a *une elasticité infinie* ssi $\exists < e_i >_{i \in N}$ phrases $\exists < L_i >_{i \in N}$ langages dans $\mathcal{CL}$ $\forall i \in N \ : \ e_i \notin L_i$ et $\{e_1, \ldots, e_n\} \subseteq L_{n+1}$

***"Powerset residuated"* semi-groupes et monoïdes.** Soit $(M,.)$ un semi-groupe (. est associatif). Soit $\mathcal{P}(M)$ l'ensemble des parties de M. Un *"powerset residuated"* semi-groupe sur $(M,.)$ est la structure $(\mathcal{P}(M), \circ, \Rightarrow, \Leftarrow, \subseteq)$ telle que pour $X,Y \subseteq M$:

$X \circ Y = \{x.y : x \in X, y \in Y\}$
$X \Rightarrow Y = \{y \in M : (\forall x \in X) x.y \in Y\}$
$Y \Leftarrow X = \{y \in M : (\forall x \in X) y.x \in Y\}$

Si $(M,.)$ est un monoïde (. est associatif, I est une unité: $\forall x \in M : I.x = x.I = x$), alors la structure ci-dessus est un *"powerset residuated"* monoïde et a pour unité $\{I\}$.

Interprétation Soit un *"powerset residuated"* semi-groupe $(\mathcal{P}(M), \circ, \Rightarrow, \Leftarrow, \subseteq)$, une interprétation est une application de types primitifs p vers des éléments $[[p]]$ dans $\mathcal{P}(M)$, étendus aux types et aux séquences de manière naturelle :

$[[C_1 \setminus C_2]] = [[C_1]] \Rightarrow [[C_2]]$
$[[C_1 \,/\, C_2]] = [[C_1]] \Leftarrow [[C_2]]$
$[[C_1 \bullet C_2]] = [[C_1]] \circ [[C_2]]$
$[[C_1, C_2, \ldots, C_n]] = [[C_1]] \circ [[C_2]]. \;\ldots\; \circ [[C_n]]$

Ces structures sont des modèles pour L selon la propriété suivante :

si $\Gamma \vdash_L C$ alors $[[\Gamma]] \subseteq [[C]]$

Si $(M,.)$ est un monoïde avec une identité I, nous ajoutons $[[\Lambda]] = \{I\}$ pour la séquence vide Λ et obtenons la propriété similaire pour $L_\emptyset$:

si $\Gamma \vdash_{L_\emptyset} C$ alors $[[\Gamma]] \subseteq [[C]]$.

3 Points limite rigides pour $L_\emptyset$

3.1 Vue d'ensemble de la construction

Definition. Nous définissons les grammaires *rigides* suivantes où p et q sont des types primitifs :

$G_{\langle 1,n \rangle} = \{a \to p \,/\, p;\;\; b \to q \,/\, q;\;\; c \to D_{\langle 1,n \rangle}\}$
où $D_{\langle 1,0 \rangle} = S$ et $D_{\langle 1,n \rangle} = (D_{\langle 1,n-1 \rangle} \,/\, (p \,/\, p)) \,/\, (q \,/\, q)$
$G_{\langle 1,* \rangle} = \{a \to p \,/\, p;\; b \to p \,/\, p;\; c \to S \,/\, (p \,/\, p)\}$

Langage. Nous obtenons (cf preuve) $\mathcal{L}(G_{\langle 1,n \rangle}) = c(b^*a^*)^n$ et $\mathcal{L}(G_{\langle 1,* \rangle}) = c(b^*a^*)^* = c\{a,b\}^*$.

Notation. Soit $\tau_{\langle 1,n \rangle}$ (et $\tau_{\langle 1,* \rangle}$) l'affectation de type aux éléments de $\{a,b,c\}$, relative à $G_{\langle 1,n \rangle}$(et à $G_{\langle 1,* \rangle}$ respectivement); ces affectations sont étendues aux éléments de $\{a,b,c\}^*$ de façon naturelle. Remarquer que ces affectations sont indépendantes de $n \geq 0$ sur $\{a,b\}^*$, nous notons alors simplement $\tau = \tau_{\langle 1,n \rangle}$ sur $\{a,b\}^*$.

Quelques points importants. Les grammaires construites ont la forme : $G_n = \{a \to A;\;\; b \to B;\;\; c \to D_n\}$; nous essayons d'expliciter ici quelques propriétés des types choisis en considérant des grammaires de cette forme.

- Nous utilisons dans cette construction des *tautologies* du calcul de Lambek autorisant les séquences vides, en particulier le schéma suivant : $C_1 \ / \ A \vdash C_1$ qui est valable lorsque A est une tautologie (c'est-à-dire vérifiant $\vdash A$) ; cela permet d'obtenir une suite de types qui déduisent les précédents $D_n \vdash D_{n-1} \vdash \ldots$...

Ceci garantit l' inclusion des langages correspondants (car si $\tau_{n-1}(w) \vdash S$ alors $\tau_n(w) \vdash S$). Mais cela ne suffit pas pour obtenir une inclusion stricte de langages; pour s'en convaincre, il suffit de considérer les grammaires de la forme ci-dessus tel que : $A = B = p \ / \ p$ (une tautologie) et $D_0 = S$, $D_n = D_{n-1} \ / \ (p \ / \ p)$, nous obtenons alors non seulement $D_n \vdash D_{n-1}$, mais aussi $D_n \vdash D_{n+1}$ pour $n > 0$, on peut montrer que dans cet exemple tous les langages $\mathcal{L}(G_n)$ pour $n > 0$ sont égaux au même langage $c\{a,b\}^*$.

- La deuxième originalité de cette construction est d'utiliser *une alternance de types* (en fait des tautologies) qui ne sont pas déductibles l'un de l'autre afin d' otenir une chaine *strictement* infinie de types ; cet aspect se retrouve dans l'usage des deux symboles a et b avec des types tels que : $\tau(a),\tau(b) \not\vdash \tau(a)$ ou $\tau(a),\tau(b) \not\vdash \tau(b)$ alors que $\tau(a),\tau(a) \vdash \tau(a)$; ce point se retrouve également dans les typages successifs du symbole c tels que :$D_{n-1} \not\vdash D_n$.

Noter que dans les langages obtenus, l'indice n joue le rôle de borne supérieure du nombre d'alternances.

- À ce stade, nous avons donc explicité une chaine strictement infinie de types. Il faut cependant remarquer que cela ne permet pas de conclure directement que les langages associés forment une chaine d'inclusion stricte. Nous illustrons ce point dans le cadre du système L (développé plus loin) avec la suite de lexiques suivants :

$\tau_n(a) = A = \tau_n(b) = B = (p \ / \ p)$ et $\tau_n(c) = D_n$ avec $D_0 = S \ / \ (p \ / \ p)$, et $D_n = D_{n-1} \bullet (p \ / \ p)$. On peut montrer que les types D_n pour $n > 0$ forment une chaine infinie de déduction stricte pour L (noter que $A,A \vdash A$ par transitivité avec $A \not\vdash A \bullet A$ dans L) mais que les langages associés pour $n > 0$ sont égaux au même langage $c\{a,b\}^*$.

Les deux points évoqués précédemment sont donc importants dans cette construction, mais n' expriment qu'une partie de son fonctionnement.

3.2 Corollaires

Pour la classe des $L_\emptyset$-grammaires rigides

- Cette construction exhibe une suite de grammaires rigides dont les langages sont strictement inclus les uns dans les autres.

Ceci montre en particulier que la classe des $L_\emptyset$-grammaires rigides a la propriété d'*élasticité infinie* (Wri89; MSW91) . Nous rappelons que pour une classe de grammaires la propriété d'élasticité finie (ie n'ayant pas l'élasticité infinie) entraîne l'apprenabilité.

Ce résultat peut surprendre dans la mesure où les classes analogues rigides (même k-valuées) des grammaires catégorielles classiques ont la propriété d'élasticité finie, cette propriété étant d'ailleurs centrale dans (Kan98).

- Nous montrons aussi que cette classe des $L_\emptyset$-grammaires rigides admet un *point limite* : $c\{a,b\}^*$; il est connu que cette propriété entraîne que la classe n'est pas aprenable à partir des exemples positifs (ici sous forme de chaines).

Autres classes

- Il faut noter tout d'abord que les grammaires rigides sont aussi des grammaires k-valuées pour $k > 1$ quelconque. Notre résultat de non-aprenabilité et de point limite s'étend donc trivialement à chaque classe k-valuée pour $k > 1$ arbitraire.

- La deuxième observation concerne une propriété des types dans la construction, ceux-ci ont en effet un degré borné, où le degré (*order*) $o(A)$ est défini par : $o(p) = 0 \quad$ si p est un type primitif ; $o(C_1 \setminus C_2) = max(o(C_1) + 1, o(C_2))$; $o(C_2 \mathbin{/} C_1) = o(C_1 \setminus C_2)$

Nous obtenons ainsi : $o(p \mathbin{/} p) = o(q \mathbin{/} q) = 1$ et $o(S) = 0$; $o(S \mathbin{/} (p \mathbin{/} p)) = 2$; $o((D_{\langle 1,n-1\rangle} \mathbin{/} (p \mathbin{/} p)) \mathbin{/} (q \mathbin{/} q)) = 2$

Notre résultat de non-aprenabilité et de point limite s'applique donc encore lorsqu'on borne le degré des types par un degré supérieur à 1.

- Enfin puisque nous n'utilisons qu'un sous-ensemble des connecteurs (pas $\setminus$), notre résultat de non-aprenabilité et de point limite s'applique aussi aux classes de grammaires dites *unidirectionnelles*.

3.3 Preuves détaillées

Dans cette partie, nous décrivons les langages reconnus par les grammaires définies précédemment. Il s'agit d'une part de justifier l'appartenance de certains mots au langage de la grammaire considérée, nous indiquons pour cela comment construire une déduction syntaxique sur les types associés. Il s'agit d'autre part de montrer qu'il n'existe pas d'autres mots reconnus, cette partie est plus délicate ; elle pourrait consister en une discussion systématique sur les formes de déduction possibles, nous avons préféré mettre au point une preuve de nature sémantique, peut-être moins directe mais qui évite une longue discussion syntaxique.

Proposition 1 (Description des langages)
$\mathcal{L}(G_{\langle 1,n\rangle}) = c(b^*a^*)^n$ *et* $\mathcal{L}(G_{\langle 1,*\rangle}) = c\{a,b\}^*$.

preuve de $c(b^*a^*)^n \subseteq \mathcal{L}(G_{\langle 1,n\rangle})$
Pour $n = 0$ c'est un axiome $\tau_{\langle 1,0\rangle}(c) = S \vdash S$.
Supposons $n > 0$ et $w' = c.w \in \mathcal{L}(G_{\langle 1,n-1\rangle})$,
- nous montrons d'abord $c.b.a.w \in \mathcal{L}(G_{\langle 1,n\rangle})$:

$$\dfrac{\dfrac{\dfrac{\vdots}{D_{\langle 1,n-1\rangle}, \tau(w) \vdash S} \quad p \mathbin{/} p \vdash p \mathbin{/} p}{D_{\langle 1,n-1\rangle} \mathbin{/} (p \mathbin{/} p), (p \mathbin{/} p), \tau(w) \vdash S} \mathbin{/} l \qquad q \mathbin{/} q \vdash q \mathbin{/} q}{\underbrace{(D_{\langle 1,n-1\rangle} \mathbin{/} (p \mathbin{/} p)) \mathbin{/} (q \mathbin{/} q)}_{= D_{\langle 1,n\rangle} = \tau_{\langle 1,n\rangle}(c)}, \underbrace{(q \mathbin{/} q)}_{=\tau(b)}, \underbrace{(p \mathbin{/} p)}_{=\tau(a)}, \tau(w) \vdash S} \mathbin{/} l$$

- nous obtenons facilement $c.w \in \mathcal{L}(G_{\langle 1,n\rangle})$ en utilisant $D_{\langle 1,n\rangle} \vdash D_{\langle 1,n-1\rangle}$ valide dans $L_\emptyset$ pour $n > 0$; (nous avons plus généralement $C_1 \;/\; (C_2 \;/\; C_2) \vdash C_1$ dans $L_\emptyset$);
- nous obtenons aussi $c.a.w \in \mathcal{L}(G_{\langle 1,n\rangle})$ à partir de $D_{\langle 1,n\rangle},p \;/\; p \vdash D_{\langle 1,n-1\rangle}$ (car $(D_{\langle 1,n\rangle} \vdash D_{\langle 1,n-1\rangle} \;/\; (p \;/\; p)$ et $D_{\langle 1,n-1\rangle} \;/\; (p \;/\; p),p \;/\; p \vdash D_{\langle 1,n-1\rangle})$
- nous obtenons ensuite $c.b.w \in \mathcal{L}(G_{\langle 1,n\rangle})$ car $D_{\langle 1,n\rangle},q \;/\; q \;=\; (D_{\langle 1,n-1\rangle} \;/\; (p \;/\; p)) \;/\; (q \;/\; q),q \;/\; q \quad \vdash D_{\langle 1,n-1\rangle} \;/\; (p \;/\; p) \vdash D_{\langle 1,n-1\rangle}$;
- enfin, ceci est étendu aux répétitions de chaque lettre a ou b séparément puisque $\tau(a),\tau(a) \vdash \tau(a)$ et $\tau(b),\tau(b) \vdash \tau(b)$ (si nous substituons chaque occurrence de a par une répétition de a ou chaque occurrence de b par une répétition de b nous obtenons encore un élément du langage $G_{\langle 1,n\rangle}$).

preuve de $\mathcal{L}(G_{\langle 1,n\rangle}) \subseteq c(b^*a^*)^n$ **(principale)** Comme expliqué auparavant, cette partie étant délicate, nous utilisons une approche sémantique permettant d' éviter ici une longue discussion syntaxique.

Nous nous plaçons dans le cadre de l'interprètation "linguistique standard" selon (Bus97) : il s'agit des structures dites "powerset residuated monoid" $(\mathcal{P}(V^*),\circ, \Rightarrow, \Leftarrow, \subseteq)$ sur le monoïde libre $(V^*,.)$ où . désigne l'opération de concaténation et V^* est l'ensemble des chaînes sur l'alphabet $V = \{a,b,c\}$.

Fixons n (arbitraire), nous définissons une interprètation par : $[[S]] = c(b^*a^*)^n$, $[[p]] = a^*$, $[[q]] = b^*$.

Supposons comme hypothèse de départ que w appartient au langage, ce qui se traduit par $\tau_{\langle 1,n\rangle}(w) \vdash S$. Par la propriété de modèles, nous avons $[[\tau_{\langle 1,n\rangle}(w)]] \subseteq [[S]]$.

Nous calculons les interprètations des types.

- Nous obtenons d'abord :$[[p \;/\; p]] = a^*$ et $[[q \;/\; q]] = b^*$
(car $[[p \;/\; p]] = \{z \in V^* : \forall x \in [[p]],z.x \in [[p]]\} = \{z \in V^* : \forall x \in a^*,z.x \in a^*\}$).
- Nous montrons maintenant par induction sur i que : $\forall i (0 \leq i \leq n) \;\; : \;\; [[D_{\langle 1,i\rangle}]] = c(b^*a^*)^{n-i}$
- cas $i = 0 \leq n$:c'est immédiat par $[[D_{\langle 1,0\rangle}]] = [[S]] = c(b^*a^*)^n$
- cas $(0 < i \leq n)$:
$[[(D_{\langle 1,i-1\rangle} \;/\; (p \;/\; p))]] = \{z \in V^* : \forall x \in [[p \;/\; p]],z.x \in [[D_{\langle 1,i-1\rangle}]]\}$
$=_{ind.} \{z \in V^* : \forall x \in a^*,z.x \in c(b^*a^*)^{(n-(i-1))}\}$
$= [[D_{\langle 1,i-1\rangle}]]$ $(= c(b^*a^*)^{(n-(i-1))})$
puis $[[D_{\langle 1,i\rangle}]] = [[(D_{\langle 1,i-1\rangle} \;/\; (p \;/\; p)) \;/\; (q \;/\; q)]]$
$= \{z \in V^* : \forall y \in [[q \;/\; q]],z.y \in [[D_{\langle 1,i-1\rangle} \;/\; (p \;/\; p)]]\}$
$= \{z \in V^* : \forall y \in b^*,z.y \in c(b^*a^*)^{(n+1-i)}\} = \{z \in V^* : \forall y \in b^*,z.y \in c(b^*a^*)^{(n-i)}.b^*.a^*\}$ (d'après ci-dessus)
$= c(b^*a^*)^{(n-i)}$ (comme souhaité)
- Nous venons de montrer en particulier que : $[[D_{\langle 1,n\rangle}]] = c$.

Nous utilisons maintenant les calculs des interprètations pour conclure. Ainsi lorsque $[[\tau_{\langle 1,n\rangle}(w)]] \subseteq [[S]]$ (conséquence de l'hypothèse de départ que w appartient au langage) ceci signifie aussi que $w = cw'$ avec $w' \in \{a,b\}^*$; nous obte-

nons $c.[[\tau_{\langle 1,n\rangle}(w')]] \subseteq c(b^*a^*)^n = [[S]]$, c'est-à-dire $[[\tau_{\langle 1,n\rangle}(w')]] \subseteq (b^*a^*)^n$ qui correspond aussi à $w' \in (b^*a^*)^n$ (car $[[\tau_{\langle 1,n\rangle}(a)]] = [[p / p]] = a^*$ et $[[\tau_{\langle 1,n\rangle}(b)]] = [[q / q]] = b^*$).
Nous obtenons donc $w \in c(b^*a^*)^n$

preuve de $c\{a,b\}^* \subseteq \mathcal{L}(G_{\langle 1,*\rangle})$
- Nous avons $c \in \mathcal{L}(G_{\langle 1,*\rangle})$ car$S / (p / p) \vdash_{L_\emptyset} S$.
- Nous obtenons ensuite $ca \in \mathcal{L}(G_{\langle 1,*\rangle})$ ainsi : $\dfrac{p / p \vdash p / p \quad S \vdash S}{S / (p / p), p / p \vdash S}$

Les autres cas sont immédiats car $p / p, p / p \vdash p / p$.

preuve de $\mathcal{L}(G_{\langle 1,*\rangle}) \subseteq c\{a,b\}^*$
Nous considérons le "powerset residuated monoid" $(\mathcal{P}(V^*), \circ , \Rightarrow , \Leftarrow , \subseteq)$ comme ci-dessus mais avec l'interpètation voisine suivante : $[[S]] = c\{a,b\}^*$, $[[p]] = a^*$, ($[[q]] = b^*$).
Supposons comme hypothèse de départ que w appartient au langage, ce qui se traduit par $\tau_{\langle 1,*\rangle}(w) \vdash S$. Par la propriété de modèles, nous avons $[[\tau_{\langle 1,*\rangle}(w)]] \subseteq [[S]]$.
- Nous calculons les interpètations des types. Nous obtenons : $[[p / p]] = a^*$ puis : $[[S / (p / p)]] = c.\{a,b\}^*$ ($= \{z \in V^* : \forall x \in [[p / p]], z.x \in [[S]]\} = \{z \in V^* : \forall x \in a^*, z.x \in c.\{a,b\}^*\}$)
- Ainsi, ayant $[[\tau_{\langle 1,*\rangle}(w)]] \subseteq [[S]] = c\{a,b\}^*$, ceci signifie aussi que $w = cw'$ avec $w' \in \{a,b\}^*$ donc w est de la forme souhaitée. ■

4 Points limite rigides pour L

Quelques points importants.
D'après la section précédente, nous savons que $D_{\langle 1,n\rangle} \vdash_{L_\emptyset} D_{\langle 1,n-1\rangle}$. Cet argument nécessaire à notre construction n'est plus valide dans L: $D_{\langle 1,n\rangle} \not\vdash_L D_{\langle 1,n-1\rangle}$. Nous allons donc transformer le lexique de la section précédente afin d'obtenir un nouveau type pour c qui permet de construire un point limite pour L avec produit. Ce nouveau lexique $\tau_{\langle 2,n\rangle}$ vérifie la condition $D_{\langle 2,n\rangle} \vdash_L D_{\langle 2,n-1\rangle}$ en utilisant la propriété de L : $(C_1 / C_2) \bullet C_2 \vdash C_1$

Un autre point important est que le nouveau type de c : $\tau_{\langle 2,n\rangle}(c)$, est dérivable dans $L_\emptyset$ à partir de $\tau_{\langle 1,n\rangle}(c)$. Ce point permet d'éviter toute une partie de la preuve en réutilisant le résultat obtenu pour $L_\emptyset$ dans la section précédente, à savoir que $\mathcal{L}(G_{\langle 1,n\rangle}) \subseteq c(b^*a^*)^n$.

Construction pour L Nous définisons les types suivants et leur affectation $\tau_{\langle 2,n\rangle}$, où $A = p / p$, $B = q / q$ et p, q sont des types primitifs :
$a \to A$; $b \to B$; $c \to D_{\langle 2,n\rangle}$
avec $D_{\langle 2,0\rangle} = S$et $D_{\langle 2,n\rangle} = (((D_{\langle 2,n-1\rangle} / A) \bullet A) / B) \bullet B$ si $n > 0$
Soit $G_{\langle 2,n\rangle}$ la grammaire rigide définie par $\tau_{\langle 2,n\rangle}$ sur l'alphabet $\{a,b,c\}$.

Soit $G_{\langle 2,*\rangle}$ la grammaire rigide, dont les types sont affectés par $\tau_{\langle 2,*\rangle}$ definie par : $G_{\langle 2,*\rangle} = \{a \to A;\ b \to A;\ c \to (S / A) \bullet A\}$

On obtient (cf preuve) $\mathcal{L}(G_{\langle 2,n\rangle}) = c(b^*a^*)^n$ et $\mathcal{L}(G_{\langle 2,*\rangle}) = c\{a,b\}^*$.

Proposition 2 (Description des langages)
$\mathcal{L}(G_{\langle 2,n\rangle}) = c(b^*a^*)^n$ et $\mathcal{L}(G_{\langle 2,*\rangle}) = c\{a,b\}^*$.

preuve de $c(b^*a^*)^n \subseteq \mathcal{L}(G_{\langle 2,n\rangle})$
Pour $n = 0$, $\tau_{\langle 2,0\rangle}(c) = S \vdash S$.
Supposons $n > 0$ et $w' = c.w \in \mathcal{L}(G_{\langle 2,n-1\rangle})$
- nous montrons tout d'abord que $c.b.a.w \in \mathcal{L}(G_{\langle 2,n\rangle})$

$$\begin{array}{c}
\vdots \\
D_{\langle 2,n-1\rangle},\tau_{\langle 2,n\rangle}(w) \vdash S \qquad A,A \vdash A \\
\hline\hline
(D_{\langle 2,n-1\rangle} \ / \ A) \bullet A,A,\tau_{\langle 2,n\rangle}(w) \vdash S \qquad B,B, \vdash B \\
\hline\hline
\underbrace{(((D_{\langle 2,n-1\rangle} \ / \ A) \bullet A) \ / \ B) \bullet B}_{=D_{\langle 2,n\rangle}=\tau_{\langle 2,n\rangle}(c)}, \underbrace{B}_{=\tau(b)}, \underbrace{A}_{=\tau(a)}, \tau_{\langle 2,n\rangle}(w) \vdash S
\end{array}$$

- nous obtenons alors $c.w \in \mathcal{L}(G_{\langle 2,n\rangle})$ puisque $D_{\langle 2,n\rangle} \vdash D_{\langle 2,n-1\rangle}$ dans L pour $n > 0$.
- nous obtenons également $c.a.w \in \mathcal{L}(G_{\langle 2,n\rangle})$ puisque $D_{\langle 2,n\rangle},A \vdash D_{\langle 2,n-1\rangle}$ ($D_{\langle 2,n\rangle} \vdash (D_{\langle 2,n-1\rangle} \ / \ A) \bullet A$ et $(D_{\langle 2,n-1\rangle} \ / \ A) \bullet A,A \vdash D_{\langle 2,n-1\rangle}$)
- nous obtenons enfin $c.b.w \in \mathcal{L}(G_{\langle 2,n\rangle})$ puisque
$D_{\langle 2,n\rangle},B = (((D_{\langle 2,n-1\rangle} \ / \ A) \bullet A) \ / \ B) \bullet B,B$
$\quad \vdash (D_{\langle 2,n-1\rangle} \ / \ A) \bullet A \vdash D_{\langle 2,n-1\rangle}$
- finalement, le résultat peut être étendu à des repetitions de a ou de b puisque $\tau(a),\tau(a) \vdash \tau(a)$ et $\tau(b),\tau(b) \vdash \tau(b)$

preuve de $\mathcal{L}(G_{\langle 2,n\rangle}) \subseteq c(b^*a^*)^n$
Nous montrons tout d'abord que $D_{\langle 1,n\rangle} \vdash_{L_\emptyset} D_{\langle 2,n\rangle}$ par induction sur n :
- le cas $n = 0$ est immédiat puisque $D_{\langle 1,0\rangle} = D_{\langle 2,0\rangle} = S$
- cas $n > 0$: nous rappellons que $A = p \ / \ p$ et $B = q \ / \ q$ sont des tautologies de $L_\emptyset$; par induction : $D_{\langle 1,n-1\rangle} \vdash_{L_\emptyset} D_{\langle 2,n-1\rangle}$, nous obtenons alors dans $L_\emptyset$:

$$\begin{array}{c}
\vdots \\
D_{\langle 1,n-1\rangle} \vdash D_{\langle 2,n-1\rangle} \qquad A \vdash A \\
\hline
D_{\langle 1,n-1\rangle} \ / \ A,A \vdash D_{\langle 2,n-1\rangle} \\
\hline
D_{\langle 1,n-1\rangle} \ / \ A \vdash D_{\langle 2,n-1\rangle} \ / \ A \qquad B \vdash B \\
\hline
(D_{\langle 1,n-1\rangle} \ / \ A) \ / \ B,B \vdash D_{\langle 2,n-1\rangle} \ / \ A \qquad \overset{\vdots}{\vdash A} \\
\hline
(D_{\langle 1,n-1\rangle} \ / \ A) \ / \ B,B \vdash (D_{\langle 2,n-1\rangle} \ / \ A) \bullet A \\
\hline
(D_{\langle 1,n-1\rangle} \ / \ A) \ / \ B \vdash ((D_{\langle 2,n-1\rangle} \ / \ A) \bullet A) \ / \ B \qquad \overset{\vdots}{\vdash B} \\
\hline
(D_{\langle 1,n-1\rangle} \ / \ A) \ / \ B \vdash ((D_{\langle 2,n-1\rangle} \ / \ A) \bullet A) \ / \ B) \bullet B
\end{array}$$

Nous pouvons dès lors finir la preuve : supposons que $\tau_{\langle 2,n\rangle}(w) \vdash_L S$, nous obtenons $\tau_{\langle 1,n\rangle}(w) \vdash_{L_\emptyset} S$ d'après la propriété précédente, donc $w \in c(b^*a^*)^n$ par la proposition 1.

preuve de $c\{a,b\}^* \subseteq \mathcal{L}(G_{\langle 2,*\rangle})$
- Nous montrons que $c \in \mathcal{L}(G_{\langle 2,*\rangle})$ puisque $(S \ / \ A) \bullet A \vdash_L S$.

- Nous montrons également que ca, ca^* et $c\{a,b\}^* \in \mathcal{L}(G_{\langle 2,*\rangle})$, puisque $A,A \vdash_L A$ (avec $A = p \,/\, p$).

preuve de $\mathcal{L}(G_{\langle 2,*\rangle}) \subseteq c\{a,b\}^*$
Cette partie suit la construction pour $L_\emptyset$ puisque $\tau_{\langle 1,*\rangle}(c) \vdash_{L_\emptyset} \tau_{\langle 2,*\rangle}(c)$, que nous montrons ci-contre :
$$\dfrac{S \,/\, A \vdash_{L_\emptyset} (S \,/\, A) \quad \begin{array}{c}\vdots\\ \vdash_{L_\emptyset} A\end{array}}{S \,/\, A \vdash_{L_\emptyset} (S \,/\, A) \bullet A}$$
donc $\tau_{\langle 1,*\rangle}(w) \vdash_{L_\emptyset} \tau_{\langle 2,*\rangle}(w)$ et $\tau_{\langle 2,*\rangle}(w) \vdash_L S$ implique $\tau_{\langle 1,*\rangle}(w) \vdash_{L_\emptyset} S$ ■

5 CONCLUSION ET REMARQUES

5.1 Non-apprenabilité de certaines classes de langages.

D'après les constructions des sections précédentes, nous déduisons en corollaire la proposition suivante :

Proposition 3 (non-apprenabilité)
La classe des langages des grammaires de Lambek rigides (ou k-valuées pour un k arbitraire) possède un point limite; la classe des grammaires de Lambek rigides (ou k-valuées pour un k arbitraire) n'est pas apprenable à partir des chaînes.

Note. Ce résultat montre que contrairement aux grammaires catégorielles classiques, les grammaires de Lambek rigides ou k-valuées ne sont pas apprenables à partir des chaînes.

Ce résultat concerne différentes variantes du calcul de Lambek :
(i) les grammaires de Lambek rigides pour $L_\emptyset$ (avec séquence vide), n'utilisant qu'un seul connecteur $/$ (unidirectionnelle);
(ii) les grammaires de Lambek rigides pour L (sans séquence vide), utilisant deux connecteurs $/$ et $\bullet$.

Notre preuve, dans ces deux cas, est une construction de points limite pour chaque classe : dans (i) et (ii), ce point limite est $c\{a,b\}^*$ qui est lui même le langage d'une grammaire de Lambek rigide comme nous l'avons vu précédemment. On remarquera que pour la première construction, les grammaires ont un ordre borné. L'extension de ces résultats aux grammaires de Lambek rigides sans produit et sans séquence vide est un travail qui a également été effectué par les auteurs et qui ne sera pas détaillé ici. Il se base sur un lexique proche de celui utilisé pour (ii), qui permet l'élimination du produit par curryfication.

5.2 Quelques remarques.

Une chaîne strictement infinie de types. Dans chaque construction, les types $D_n = D_{\langle i,n\rangle}$ de c sont tels que : ... $\vdash D_n \vdash D_{n-1} \vdash \ldots \vdash D_0$. C'est un point essentiel pour l'inclusion des langages.

De plus, $D_{n-1} \not\vdash D_n$, ce qui est nécessaire à l'inclusion stricte des langages. Notre construction est donc basée sur une chaîne strictement infinie de types, où chaque type déduit les précédents, et ceci pour les différentes variantes du calcul de Lambek considérées.

Afin d'obtenir une inclusion stricte, nous utilisons de plus une alternance de types non déductibles les uns des autres (ici les types de a et b).

Nous avons donc explicité une chaine strictement infinie de types; mais comme nous l'avons déjà expliqué cette seule propriété ne permet pas de conclure directement que les langages associés forment une chaine d'inclusion stricte : cette propriété n'exprime qu'une partie du fonctionnement de notre construction.

Extension à d'autres classes de grammaires. Nous avons présenté deux formes de grammaires, chacune décrivant l'existence d'un point limite pour la classe de grammaires à laquelle elle appartient. Il est possible d'étendre ces résultats à d'autres classes de grammaires par simple examen des règles utilisées lors des preuves précédemment détaillées. Nous pouvons en effet constater que la seule utilisation des règles d'introductions à droite de $\backslash$ et $/$ pour la preuve concernant les grammaires rigides de Lambek sans séquence vide, est la transitivité. Ceci implique immédiatement que les seules règles d'introduction à gauche de $\backslash$ et $/$ associées aux règles de transitivité (également valides dans les grammaires catégorielles de Steedman (Ste88))[4] et aux règles d'introduction du produit suffisent pour obtenir un point limite.

Perspectives. Nous nous intéressons également à d'autres variantes des grammaires catégorielles, comme les grammaires de Lambek non-associatives ou les grammaires catégorielles multi-modales, pour lesquelles la construction de *points limite rigides* semble s'étendre. Nous comptons également considérer des langages structurés (langages d'arbres) au lieu des langages de chaînes pour l'inférence grammaticale. D'après (BR01), les grammaires de Lambek rigides sont apprenables à la limite à partir de phrases structurées. De telles structures, étudiées dans (Tie99), sont les structures des arbres de preuves complètes sous formes normales. Notre résultat montre que sans structure, certaines classes de grammaires de Lambek ne sont pas apprenables à la limite à partir d'exemples positifs. Il serait maintenant très intéressant de trouver une structure intermédiaire, plus réaliste linguistiquement que les structures de preuves complètes mais qui préserve l'apprenabilité à la limite des grammaires de Lambek.

Remerciements. Nous remercions les membres de l'ARC Gracq (action de recherche coopérative de l'Inria) et les rapporteurs anonymes pour leur aide.

RÉFÉRENCES

Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

4. On rappelle que ces règles de transitivité peuvent être formulées par les règles suivantes: $A \backslash B, B \backslash C \Rightarrow A \backslash C$ et $C / B, B / A \Rightarrow C / A$

Y. Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

Roberto Bonato and Christian Retoré . Learning rigid lambek grammars and minimalist grammars from structured sentences. *Third workshop on Learning Language in Logic, Strasbourg*, september 2001.

W. Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen (vBtM97), chapter 12, pages 683–736.

Claudia Casadio. Semantic categories and the development of categorial grammars. In Oehrle et al. (OBW88), pages 95–124.

D. Dudau-Sofronie, I. Tellier and M. Tommasi. Learning categorial grammars from semantic types. In *13e Amsterdam Colloquium*, Palaiseau, décembre 2001.

Jean-Yves Girard. Linear logic: its syntax and semantics. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*, pages 1–42. Cambridge University Press, 1995.

E.M. Gold. Language identification in the limit. *Information and control*, 10:447–474, 1967.

Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, 65:154–169, 1958.

Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen (vBtM97), chapter 2, pages 93–177.

Tatsuya Motoki, Takeshi Shinohara, and Keith Wright. The correct definition of finite elasticity: Corrigendum to identification of unions. In *The fourth Annual Workshop on Computational Learning Theory*, page 375, San Mateo, Calif., 1991. Morgan Kaufmann.

Jacques Nicolas. Grammatical inference as unification. Rapport de Recherche RR-3632, INRIA, 1999. http://www.inria.fr/RRRT/publications-eng.html.

R. Oehrle, E. Bach, and D. Wheeler, editors. Dordrecht, 1988. Reidel.

Mati Pentus. Lambek grammars are context-free. In *Logic in Computer Science*. IEEE Computer Society Press, 1993.

Christian Retoré. Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles. *T.S.I*, 2000.

T. Shinohara. Inductive inference from positive data is powerful. In *The 1990 Workshop on Computational Learning Theory*, pages 97–110, San Mateo, California, 1990. Morgan Kaufmann.

Mark Steedman. Combinators and grammars. In Oehrle et al. (OBW88).

Hans-Jörg Tiede. *Deductive Systems and Grammars: Proofs as Grammatical Structures*. PhD thesis, Illinois Wesleyan University, 1999.

J. van Benthem and A. ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

K. Wright. Identifications of unions of languages drawn from an identifiable class. In *The 1989 Workshop on Computational Learning Theory*, pages 328–333, San Mateo, Calif., 1989. Morgan Kaufmann.

C. Gaifman Yehoshua Bar-Hillel and E. Shamir. On categorial and phrase-structure grammars. *Bulletin of the Research Council of Israel*, 9F:1–16, 1960.