Apprentissage des grammaires de Lambek rigides et d'arité bornée pour le traitement automatique des langues.

Denis Béchet[1], Annie Foret[2]

[1] IRISA, INRIA. Denis.Bechet@irisa.fr

[2] IRISA, Université de Rennes 1. Annie.Foret@irisa.fr
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex
France

Résumé : Nous nous intéressons au problème de l'apprentissage des grammaires catégorielles utilisées pour la modélisation syntaxique des langues. Récemment, des algorithmes ont été proposés dans le modèle d'apprentissage de Gold, pour certaines classes de grammaires catégorielles. En revanche, les grammaires de Lambek rigides ou k-valuées ne sont pas apprenables à partir des chaînes. Nous nous intéressons ici à une autre classification de ces grammaires selon l'*arité des types*. Nous montrons ainsi qu'une borne sur l'arité des types permet d'identifier à la limite les grammaires de Lambek rigides ou k-valuées dans le cas non associatif.

Ces résultats répondent à certaines questions de linguistique mathématique et d'inférence grammaticale et indiquent une direction dans laquelle poursuivre l'étude de l'acquisition automatique de ressources lexicales.

Mots clef Apprentissage – Inférence grammaticale – Traitement automatique des langues – Linguistique informatique – Grammaires catégorielles de Lambek – Modèle de Gold

1 Introduction

Nous nous intéressons au problème de l'apprentissage des grammaires catégorielles, importantes pour le traitement automatique des langues en particulier pour la modélisation syntaxique des langues.

Les grammaires lexicalisées offrent divers avantages dans un cadre d'apprentissage : lorsqu'un mot nouveau est rencontré, il suffit de compléter les informations pour ce mot, sans changer les règles de la grammaire.

Le modèle de l'apprentissage (au sens de Gold (Gold, 1967)) qui est utilisé 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 de V^+ vers $\mathcal{G}$ telle que pour tout $G \in \mathcal{G}$ 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 (Gold, 1967). Le regain d'intérêt pour l'apprentissage à la limite à partir d'exemples positifs est dû aux travaux de Dana Angluin (Angluin, 1980), 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 (Shinohara, 1990) qui montra que les grammaires contextuelles avec au plus k règles sont également apprenables. Les résultats d'apprentissage de Kanazawa (Kanazawa, 1998) viennent compléter la liste des grammaires non triviales apprenables avec les grammaires catégorielles k-valuées classiques dont l'apprentissage se révèle particulièrement intéressant du fait de leur complète lexicalisation.

Depuis les premiers articles d'Ajdukiewicz (Ajdukiewicz, 1935) et Bar-Hillel (Bar-Hillel, 1953) définissant les grammaires catégorielles classiques (dites AB), divers systèmes lexicalisés sont apparus. Nous nous intéressons ici aux grammaires catégorielles de Lambek (Lambek, 1958) qui en sont un raffinement. Ce modèle de la syntaxe des langues entretient des rapports étroits avec la sémantique (à la Montague).

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 (Girard, 1995).

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 dans (Kanazawa, 1998) et (Nicolas, 1999). Acquérir une grammaire revient dans ce cadre à acquérir les types associés aux mots. De tels algorithmes d'acquisition d'une 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 sont ainsi devenus depuis peu un thème de recherche très actif. Le lien avec la sémantique est aussi utilisé dans cette perspective, par exemple dans (Dudau-Sofronie *et al.*, décembre 2001).

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 (Pentus, 1993) justifie 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 (Yehoshua Bar-Hillel & Shamir, 1960), 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 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 (Bonato & Retoré, september 2001).

En fait, contrairement aux grammaires catégorielles classiques, les grammaires de Lambek rigides ou k-valuées ne sont pas apprenables à partir des chaînes (Foret & Le Nir, 2002a; Foret & Le Nir, 2002b). Aussi, nous nous intéressons ici à une autre classification de ces grammaires selon l'arité des types. Nous montrons ainsi qu'une borne sur l'arité des types permet d'identifier à la limite les grammaires de Lambek rigides ou k-valuées, dans le cas non associatif. Cette variante introduite dans (Lambek, 1961), sert en fait de base à une hiérarchie de systèmes grammaticaux catégoriels et se situe entre les grammaires catégorielles classiques et les grammaires de Lambek associatives, (lorsque l'on considère le même lexique, mais dans les différents calculs). Globalement, la méthode pour démontrer l'apprenabilité de ces langages repose sur la possibilité de construire progressivement, à partir des exemples de phrases correctes, un tableau de valeurs booléennes caractérisant la grammaire que l'on est en train d'apprendre.

L'article est organisé comme suit. La section 2 présente les définitions nécessaires et les résultats connus. La section 3 donne notre résultat d'apprenabilité avec la preuve pour le calcul de Lambek non associatif et une présentation rapide de l'algorithme d'apprentissage inférant la présentation sous forme de tableau, équivalente au système logique de Lambek à inférer, pour le cas de l'apprentissage à partir de structures. La section 4 conclut et donne quelques perspectives pour la classe des grammaires d'arité bornée abordée dans l'article.

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 consulter les articles originaux de Lambek (Lambek, 1958; Lambek, 1961).

2.1.1 Types primitifs.

Pour décrire des types syntaxiques associés aux mots (de phrases) nous commençons par introduire des types primitifs : P pour le type des phrases (bien construites) et GN pour le type des groupes nominaux 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 GN, par exemple “pauvre Pierre” ou “fresh milk”.

2.1.2 Types composés.

Nous pouvons aussi introduire des types composés : en général une expression de type x/y suivie par une expression de type y produit une expression de type x ; de façon semblable, une expression de type $y\backslash x$ 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”, le type composé GN/GN. Un verbe intransitif tel que *travaille* dans “Pierre travaille” est associé au type $GN\backslash P$. Un verbe transitif tel que *aime* est associé au type $(GN\backslash P)/GN$.

2.1.3 Les pronoms et la montée de type.

Dans des exemples tels que “il travaille” ou “il le mange” le pronom “il” est associé au type $P/(GN\backslash P)$ et le pronom “le” est associé au type $(GN\backslash P)/((GN\backslash P)/GN)$. Ce codage met en valeur la fonction des pronoms qui passent du rôle de groupe nominal a celui de modificateur de verbe.

Dans le cas du pronom “il”, une expression de type GN 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).

2.1.4 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 en fait sur une conception trop strictement concaténative.

Par contre, les grammaires de Lambek peuvent parfois paraître trop synthétiques ce qui aboutit à des phrases utilisant des constructions abusives comme par exemple l’utilisation des conjonctions de coordination sur des types trop généraux. Ainsi, si la conjonction de coordination “et” est typée avec $X/(X\backslash X)$ pour tout X, la phrase “il le regarde et prend” est considérée comme correcte d’un point de vue logique en utilisant le type des verbes transitifs $(GN\backslash P)/GN$ pour X. La phrase semble toutefois vraiment bizarre.

2.1.5 La variante non associative du calcul de Lambek.

Ce genre de considérations ainsi que les résultats de non apprenabilité des grammaires de Lambek, nous ont conduit à étudier la variante non associative du calcul de Lambek dans l’optique de l’apprentissage. Ce calcul se place à la fois comme une alternative logique des grammaires AB et comme un système plus contraint que la version

associative mais qui autorise des constructions très intéressantes dues en particulier à la montée des types.

Par exemple, au lieu de coder les pronoms sujets comme des modificateurs de verbe, puisqu'ils se placent au même endroit que le groupe nominal qu'ils remplacent, nous pouvons utiliser un type $\overline{GN} = X/(GN\backslash X)$ qui code le fait que le pronom sujet peut être remplacé par un groupe nominal GN. Dans ce cadre, un verbe intransitif se code avec le type $\overline{GN}\backslash P$ et la preuve logique de $GN \vdash \overline{GN}$ permet d'utiliser GN à la place de $\overline{GN}$. Par contre, comme $\overline{GN} \nvdash GN$, nous ne pouvons pas remplacer dans tous les cas un nom par un pronom. Cela est uniquement possible là où un $\overline{GN}$ est attendu. Avec ce codage simple, nous pouvons définir des hiérarchies complexes de types (comme coder des ordres quelconques) sans avoir les inconvénients qui arriveraient avec la version associative. En effet, la non associativité interdit les interactions entre deux montées de types présentes dans une phrase ce qui n'est malheureusement pas le cas pour le calcul de Lambek associatif. Dans ce cas, la version associative donnerait comme correctes des phrases comportant des erreurs syntaxiques.

2.2 Définitions formelles

Nous notons de façon générale V^* l'ensemble des mots (comprenant le mot vide) sur un alphabet V et V^+ l'ensemble des mots non vides sur l'alphabet V. De même, nous notons $\mathcal{T}_V$ l'ensemble des arbres binaires dont les feuilles appartiennent à V. Par la suite, Σ désigne un alphabet donné qui pourra représenter les mots d'une langue naturelle.

2.2.1 Types.

Les *types (ou catégories)* sont construits à partir de Pr (ensemble des *types primitifs*) et de deux connecteurs binaires $/$ et $\backslash$. Tp représente l'ensemble des types.

$$Tp ::= Pr \mid Tp\backslash Tp \mid Tp/Tp$$

D'un point de vue logique, les connecteurs $/, \backslash$ sont des formes d'implications (linéaires). Pr contient un type particulier, noté P, également appelé *type principal* qui est associé aux phrases correctes du langage.

2.2.2 Grammaires catégorielles.

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

2.2.3 Séquents.

Dans le cadre des grammaires catégorielles non associatives, un séquent noté $\Gamma \vdash C$ est une paire (Γ, C) où Γ est un arbre binaire de types (potentiellement vide) et $C \in Tp$ est un type. Γ est appelé l'antécédent et C la conclusion du séquent. L'ensemble des arbres est noté $\mathcal{T}_{Tp}$.

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 non associatif ou les grammaires AB par exemple). Nous commençons par présenter les règles de dérivation pour les grammaires AB.

2.2.4 Dérivation $\vdash_{AB}$

La relation $\vdash_{AB}$ est la plus petite relation $\vdash$ entre $\mathcal{T}_{Tp}$ et Tp, telle que pour tout $\Gamma, \Delta \in \mathcal{T}_{Tp}$ et pour tout $A, B \in Tp$:

$A \vdash A$
si $\Gamma \vdash A$ et $\Delta \vdash A \backslash B$ alors $(\Gamma, \Delta) \vdash B$ (“Application en arrière $\backslash_e$ ”)
si $\Gamma \vdash B/A$ et $\Delta \vdash A$ alors $(\Gamma, \Delta) \vdash B$ (“Application en avant $/_e$ ”)

Nous présentons maintenant une formulation du calcul de Lambek non associatif, noté NL, n’incluant pas le produit, constitué de règles d’introduction à gauche et à droite d’un séquent.

2.2.5 Dérivation dans Lambek $\vdash_{NL}$.

La relation $\vdash_{NL}$ entre $\mathcal{T}_{Tp}$ et Tp, dans le cas non associatif peut être vue comme une extension des règles pour AB, en lui conférant des propriétés de nature logique (et permettant une interface avec la sémantique). Nous utilisons une présentation dans le style de Gentzen.

Un contexte $\Gamma[\cdot]$ est un arbre binaire avec un trou. Pour une formule ou un arbre binaire de formules X, $\Gamma[X]$ est l’arbre binaire obtenue à partir de $\Gamma[\cdot]$ en remplissant le trou par X.

Un séquent est valide dans NL (noté $\Gamma \vdash_{NL} A$) ssi $\Gamma \vdash A$ peut se déduire des règles suivantes :

$$\frac{}{A \vdash A}\,Ax \qquad \frac{(\Gamma, B) \vdash A}{\Gamma \vdash A/B}/R \qquad \frac{(A, \Gamma) \vdash B}{\Gamma \vdash A \backslash B}\backslash R$$

$$\frac{\Gamma \vdash A \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash B}\,Cut \qquad \frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(B/A, \Gamma)] \vdash C}/L \qquad \frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(\Gamma, A \backslash B)] \vdash C}\backslash L$$

2.2.6 Élimination des coupures.

Nous rappelons que la règle de coupure peut être éliminée dans $\vdash_{NL}$: chaque séquent dérivable admet une dérivation sans coupure.

2.2.7 Production.

Si T est un arbre dont les feuilles sont des éléments d’un ensemble X, $prod_X(T) \in X^+$ est la liste des feuilles de T. Cette notation est utilisée pour les arbres binaires de mots $prod_\Sigma$, et pour les arbres binaires de formules $prod_{Tp}$.

2.2.8 Langage.

Soit G une grammaire catégorielle sur Σ avec P pour type principal.

• *G génère un arbre binaire de mots $T \in \mathcal{T}_\Sigma$* ssi il existe Γ un arbre binaire de types, $c_1, \ldots, c_n \in \Sigma$ et $A_1, \ldots, A_n \in Tp$ tels que :

$$\begin{cases} prod_\Sigma(T) = c_1 \cdots c_n \\ G : c_i \mapsto A_i \text{ pour } 1 \leq i \leq n \\ \Gamma = T[c_1 \to A_1, \ldots, c_n \to A_n] \\ \Gamma \vdash_{NL} P \end{cases}$$

où $T[c_1 \to A_1, \ldots, c_n \to A_n]$ désigne l'arbre binaire obtenu à partir de T en substituant de gauche à droite les occurrences de $c_1, \ldots, c_n$ par $A_1, \ldots, A_n$.

• *G génère une chaîne $c_1 \cdots c_n \in \Sigma^+$* ssi il existe $T \in \mathcal{T}_\Sigma$ tel que $prod_\Sigma(T) = c_1 \cdots c_n$ et G génère T.

• Le langage de G, noté $\mathcal{L}_{NL}(G)$ est l'ensemble des chaînes générées par G.

• Nous définissons de la même manière $\mathcal{L}_{AB}(G)$ en remplaçant $\vdash_{NL}$ par $\vdash_{AB}$.

2.2.9 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.

2.2.10 Exemple.

Soient $\Sigma_1 = \{Pierre, Marie, aime\}$ et $Pr = \{P, GN\}$ pour le type des phrases et des noms respectivement. Avec $G_1 = \{Pierre \mapsto GN,\ Marie \mapsto GN,\ aime \mapsto GN \backslash (P/GN)\}$, nous obtenons $(Pierre\ aime\ Marie) \in \mathcal{L}_{NL}(G_1)$ car $((GN, GN \backslash (P/GN)), GN) \vdash_{NL} P$)[1]. G_1 est une grammaire rigide (ou 1-valuée).

2.3 Inférence de grammaires catégorielles

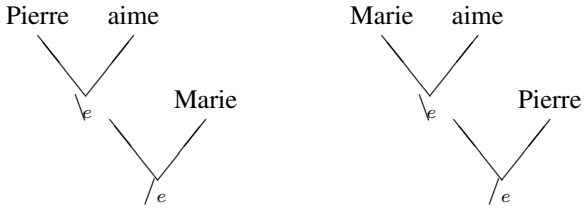
Nous illustrons ici l'inférence grammaticale de grammaires catégorielles. Nous considérons le cadre le plus simple des grammaires AB rigides, avec des exemples positifs supposés structurés en arbres étiquetés qui représentent la structure en constituants avec des étiquettes sur les nœuds internes indiquant le sens de l'application (application en avant $/_e$ ou en arrière $\backslash_e$[2]). 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 (Kanazawa, 1998) pour certaines classes de grammaires AB. Nous donnons ci-dessous un exemple illustrant l'algorithme.

[1] Voici quelques détails : $P \vdash P$ et $GN \vdash GN$ donnent $((P/GN), GN) \vdash P$ par $/L$, ce qui donne ensuite avec $GN \vdash GN$ la conclusion $((GN, GN \backslash (P/GN)), GN) \vdash P$ par $\backslash L$.

[2] l'indice e rappelle qu'il s'agit d'élimination

2.3.1 Exemple

Soient les deux exemples structurés suivants :



Une première phase consiste à générer des types arguments, la racine des arbres étant étiquetée par P. Une variable est introduite pour chaque argument apparaissant dans les structures. Deux variables sont introduites pour la première phrase, X_1 et X_2 et deux autres variables pour la seconde phrase, X_3 et X_4. Suit une phase de génération des types fonctions qui propagent les variables sur les entrées lexicales. La dernière phase est l'unification des types obtenus. Ceci est résumé dans le tableau suivant :

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

3 Apprenabilité des Grammaires d'arité bornée

3.1 Définitions

Nous reprenons ici une notation pratique et une définition d'arité[3] selon (Kanazawa, 1998).

• Chaque type A peut s'écrire de façon unique sous la forme suivante :

$$p|A_n|A_{n-1}|\cdots|A_1$$

où p est primitif et $|$ désigne $\backslash$ ou bien $/$. Les types arguments se placent alors toujours à droite contrairement à la notation de Lambek.

• Le nombre n est appelé *l'arité de* A.

• Les types $|A_i$ sont les *sous-types arguments de* A. Nous appelons $|A_i$ le i-ème sous-type argument. Pour une formule d'arité n, il y a n sous-types arguments qui comportent chacun un sens d'application (en avant ou en arrière).

• Le j-ème sous-type principal (ou de tête) désigne $p|A_n|\cdots|A_j$. Le type p est la tête de A (dénotée $tête(A)$) et par convention représente le dernier sous-type principal. Pour une formule d'arité n, il y a donc $n+1$ sous-type principaux.

Par exemple, le type $(GN \setminus P)/\!/(GN/GN)$ s'écrit $P|\backslash GN|/(GN/GN)$. Son arité est 2. Les sous-types arguments sont (GN/GN) orienté $/$ et GN orienté $\backslash$. Les sous-types principaux sont $P\backslash GN/(GN/GN)$, $P\backslash GN$ et P. Nous étendons cette définition aux grammaires catégorielles :

[3]Cette notion d'arité est naturelle. Elle était déjà connue mais n'était pas utilisée comme critère pour l'apprenabilité.

• Une grammaire G est dite *d'arité* n si tous les types qu'elle associe aux mots sont eux-mêmes d'arité au plus n ; G sera dite *d'arité exactement* n si de plus, au moins un de ces types est d'arité n.

• La classe $\mathcal{L}_1^{(arité \leq n)}$ désigne la classe des langages associés aux grammaires rigides d'arité n.

• La classe $\mathcal{L}_k^{(arité \leq n)}$ désigne la classe des langages associés aux grammaires k-valuées d'arité n.

• La classe $\mathcal{L}^{(arité \leq n)}$ désigne la classe des langages associés aux grammaires d'arité n.

Dans le but de caractériser les langages de grammaires rigides d'arité n, nous associons à une grammaire G d'une part une grammaire généralisée G_{var} (avec des variables au lieu des arguments) et un tableau noté $Tab(G, n)$ servant à exprimer des contraintes de déduction entre les types et sous-types principaux apparaissant dans G.

Soit $\Sigma = \{c_1, \ldots c_q\}$ un alphabet fixé, de taille q, et G une grammaire rigide d'arité n de la classe $\mathcal{L}_1^{(arité \leq n)}$. Nous lui associons :

• la grammaire généralisée G_{var} obtenue en remplaçant dans le type associé au i-ème mot de Σ son symbole de tête par la variable x_{i,n_i} avec n_i l'arité du type plus 1 et chaque j-ème argument (de la droite vers la gauche dans le type) par la variable $x_{i,j}$;

• le tableau[4] $Tab(G, n)$ qui est un tableau carré de dimension $q \times (n + 1)$, dont les cellules sont de valeur indéfinie ($\perp$) ou boolénne (V ou F) tel que :

– la colonne (i', j') représente le j'-ème sous-type argument du type associé au i'-ème mot de Σ (s'il existe, indéfini sinon — en particulier la dernière colonne de chaque mot dans le tableau est toujours indéfinie car il n'y a qu'au plus n sous-types arguments) ;
– la ligne (i, j) représente le j-ème sous-type principal de tête du type associé au i-ème mot de Σ (s'il existe, indéfini sinon) ;
– la cellule $(i, j), (i', j')$ sert à indiquer la relation de déduction entre le sous-type argument $x_{(i',j')}$ et le sous-type principal $x_{(i,n_i)}|\cdots|x_{(i,j)}$: elle indique si le j-ème sous-type principal du type associé au i-ème mot de Σ déduit le j'-ème sous-type argument du type associé au i'-ème mot de Σ (s'ils existent, indéfini sinon). Cette valeur est calculée par un démonstrateur pour le système NL qui est décidable en temps polynomial.

Ce tableau sert à décrire des grammaires génériques d'arité n pour Σ, de forme :

$$\{c_i \mapsto x_{i,0}|x_{i,1}|x_{i,2}|\cdots|x_{i,n_i}\}$$

et des contraintes de déduction entre sous-formules (avec des variables distinctes).

[4] Cette présentation n'est pas optimisée, mais plus uniforme.

3.1.1 Exemple

Prenons l'exemple de la grammaire $G_2 = \{Pierre \mapsto GN,\ Marie \mapsto GN,\ il \mapsto X|_{/}(GN\backslash X),\ aime \mapsto P|\backslash\overline{GN}|_{/}GN\}$ avec $\overline{GN} = X/(GN\backslash X)$[5] [6]. Cette grammaire comporte 4 mots et est d'arité 2. Le tableau aura donc $4 \times (2+1) = 12$ lignes et colonnes :

		$Pierre$			$Marie$			il			$aime$		
		$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\|_{/}(GN\backslash X)$	$\bot$	$\bot$	$\|_{/}GN$	$\backslash\overline{GN}$	$\bot$
$Pierre$	GN	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	V	V	$\bot$
	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$Marie$	GN	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	V	V	$\bot$
	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
il	$\overline{GN}$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	F	V	$\bot$
	X	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	F	F	$\bot$
	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$aime$	$P\|\backslash\overline{GN}\|_{/}GN$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	F	F	$\bot$
	$P\|\backslash\overline{GN}$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	F	F	$\bot$
	P	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	F	$\bot$	$\bot$	F	F	$\bot$

Ici, la relation de déduction n'est pas l'identité pour le cas $GN \vdash_{NL} \overline{GN}$ qui est vrai dans NL mais pas dans AB.

3.1.2 Remarque.

Cette définition s'étend facilement aux grammaires k-valuées en multipliant le nombre de colonnes par k. Pour la suite, nous supposerons que la grammaire est rigide mais la généralisation aux grammaires k-valuées est aisée.

3.2 Propriétés

3.2.1 Lemme fondamental

Soient G et G' deux grammaires rigides d'arité n : si (i) ces grammaires ont même grammaire généralisée et si (ii) leurs tableaux associés coïncident, alors (iii) ces grammaires ont le même langage.

Nous notons $G \sim G'$ lorsque ces grammaires vérifient (i) et (ii) ci-dessus. Ce lemme exprime donc que $G \sim G'$ implique $\mathcal{L}(G) = \mathcal{L}(G')$.

La démonstration de ce lemme est donné en section 3.2.3. Elle utilise une standardisation des preuves de NL présentée en section 3.2.2.

[5] Avec la notation fonctionnelle des types présentée au début de la section 3.

[6] Noter que vis à vis du tableau X se comporte plutôt comme un type primitif.

3.2.2 Une présentation alternative pour NL

Pour effectuer la preuve du lemme, nous utilisons un système intermédiaire appelé NLD_0^{**}. Aarts et Trautwein (Aarts & Trautwein, 1995) ont montré l'équivalence entre NL et ce système appelé NLD_0^{**} :

$$\frac{}{A \vdash A}\,Ax \qquad \frac{(C,B) \vdash A}{C \vdash A/B}/R^* \qquad \frac{(A,C) \vdash B}{C \vdash A\backslash B}\backslash R^*$$

$$\frac{D \vdash C \quad \Delta[B] \vdash A}{\Delta[(B/C,D)] \vdash A}/L^* \qquad \frac{D \vdash C \quad \Delta[B] \vdash A}{\Delta[(D,C\backslash B)] \vdash A}\backslash L^*$$

3.2.3 Démonstration du lemme.

En considérant le système NLD_0^{**}, on montre par récurrence sur la longueur de déduction que si $G \sim G'$ et $\Gamma \vdash P$ (où P est atomique) avec Γ ne contenant que des *sous-formules de tête* des types de G alors on obtient une déduction (semblable) pour les types de G' en remplaçant chaque j-ème argument du i-ème type dans G par le j-ème argument du i-ème type dans G'.

3.2.4 Corollaire [Apprenabilité]

La classe des langages des grammaires rigides d'arité n est identifiable à la limite dans le modèle de Gold.

3.2.5 Démonstration du corollaire.

Soit Σ et n fixés :

- d'une part, le nombre de grammaires généralisées est fini ;
- d'autre part, il existe un nombre fini de tableaux associés pour une arité donnée.

D'après le lemme fondamental, ceci montre que le nombre de langages rigides pour une arité donnée est fini[7] et nous pouvons ensuite appliquer les critères généraux d'apprenabilité qui stipulent que les classes de langage dont on peut borner la taille en fonction de la taille du lexique est apprenable, puisqu'elle possède la propriété de *densité finie* et donc d'élasticité finie (Wright, 1989; Motoki *et al.*, 1991; Shinohara, 1990; Shinohara, 1991). Ce résultat très fort permet aussi de passer au cas des grammaires k-valuées et à l'apprentissage sur des chaînes plutôt que sur des structures.

3.3 Vers un algorithme d'apprentissage

En fait, la démonstration nous donne un algorithme pour construire facilement une représentation equivalente de la grammaire que l'on est en train d'apprendre. Cet algorithme prend en argument une liste de phrases structurées comme pour le cas de

[7] À ne pas confondre avec le nombre de grammaires qui est non borné.

l'apprentissage des grammaires AB de la section 2.3 et place progressivement les valeurs V du tableau présenté ci-dessous en fonction des phrases qui lui sont présentées. Au bout d'un certain nombre d'exemples, ce tableau converge (plus aucune nouvelle valeur ne vient remplacer les valeurs initiales $\perp$ placées dans toutes les cellules). Il est alors caractéristique de la partie utile de la grammaire apprise sans toutefois nous donner directement une grammaire de Lambek compatible (il faudrait pour cela inférer des types pour chacun des mots du lexique à partir des valeurs du tableau).

4 Discussion et perspectives

Ainsi, nous avons montré que la classe des langages qui correspondent aux grammaires rigides du calcul de Lambek non associatif d'arité bornée par une constante n possède la propriété d'élasticité finie. En utilisant des résultats généraux sur ce type de classes, nous en déduisons que pour tout k, la classe $\mathcal{G}_k^{(arité \leq n)}$ est apprenable à la fois sur des exemples structurés et sur des chaînes.

Du point de vue algorithmique, le mécanisme pour la démonstration de l'élasticité finie dans le cas des grammaires rigides avec comme entrées des phrases structurées tente de remplir un tableau indiquant les relations deux à deux entre un mot pris en tant que fonction partiellement appliquée et un des paramètres d'un autre mot. La relation est plus générale que l'identité (d'où la différence avec les grammaires AB) puisqu'elle est basée sur la relation de prouvabilité dans NL. Ceci ne fournit pourtant pas directement un algorithme efficace d'apprentissage car le tableau obtenu n'est pas assez synthétique. Il donne en effet la liste exhaustive des relations binaires entre les mots du lexique et pas une grammaires de Lambek. Cet aspect pourra faire l'objet d'études ultérieures.

Une autre perspective nous paraît intéressante à explorer en partant de la vision sous forme de tableau de composition : le degré de précision qui sans être prohibitif (il synthétise les exemples en entrées par des relations binaires) donne une bonne approximation des grammaires ; et cela nous offre la possibilité d'introduire une analyse stochastique en remplaçant les valeurs $\perp$ et V du tableau par des probabilités ce qui permettrait vraisemblablement de rendre l'algorithme robuste au bruit.

Des questions d'inclusion ou de capacité générative relatives à la hiérarchie de langages induite par l'arité restent aussi à explorer.

Références

AARTS E. & TRAUTWEIN K. (1995). Non-associative Lambek categorial grammar in polynomial time. *Mathematical Logic Quaterly*, **41**, 476–484.

AJDUKIEWICZ K. (1935). Die syntaktische konnexität. *Studia Philosophica*, **1**, 1–27.

ANGLUIN D. (1980). Inductive inference of formal languages from positive data. *Information and Control*, **45**, 117–135.

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

BONATO R. & RETORÉ C. (september 2001). Learning rigid lambek grammars and minimalist grammars from struc tured sentences. *Third workshop on Learning Language in Logic, Strasbourg*.

DUDAU-SOFRONIE, TELLIER & TOMMASI (décembre 2001). Learning categorial grammars from semantic types. In *13e Amsterdam Colloquium*, Palaiseau.

FORET A. & LE NIR Y. (2002a). Lambek rigid grammars are not learnable from strings. In *COLING'2002, 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

FORET A. & LE NIR Y. (2002b). On limit points for some variants of rigid lambek grammars. In *ICGI'2002, the 6th International Colloquium on Grammatical Inference*, number 2484 in Lecture Notes in Artificial Intelligence : Springer-Verlag.

GIRARD J.-Y. (1995). Linear logic : its syntax and semantics. In J.-Y. GIRARD, Y. LAFONT & L. REGNIER, Eds., *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*, p. 1–42 : Cambridge University Press.

GOLD E. (1967). Language identification in the limit. *Information and control*, **10**, 447–474.

KANAZAWA M. (1998). *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI. distributed by Cambridge University Press.

LAMBEK J. (1958). The mathematics of sentence structure. *American mathematical monthly*, **65**, 154–169.

LAMBEK J. (1961). On the calculus of syntactic types. In R. JAKOBSON, Ed., *Structure of language and its mathematical aspects*, p. 166–178. American Mathematical Society.

MOTOKI T., SHINOHARA T. & WRIGHT K. (1991). The correct definition of finite elasticity : Corrigendum to identification of unions. In *The fourth Annual Workshop on Computational Learning Theory*, p. 375, San Mateo, Calif. : Morgan Kaufmann.

NICOLAS J. (1999). *Grammatical inference as unification*. Rapport de Recherche RR-3632, INRIA. http ://www.inria.fr/RRRT/publications-eng.html.

PENTUS M. (1993). Lambek grammars are context-free. In *Logic in Computer Science* : IEEE Computer Society Press.

SHINOHARA T. (1990). Inductive inference from positive data is powerful. In *The 1990 Workshop on Computational Learning Theory*, p. 97–110, San Mateo, California : Morgan Kaufmann.

SHINOHARA T. (1991). Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, **8 (4)**, 371–384. Special Issue on Algorithmic Learning Theory for ALT'90.

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

YEHOSHUA BAR-HILLEL C. G. & SHAMIR E. (1960). On categorial and phrase-structure grammars. *Bulletin of the Research Council of Israel*, **9F**, 1–16.