k-valued Non-Associative Lambek Categorial Grammars are not Learnable from Strings

Denis Béchet
INRIA, IRISA
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex
France
Denis.Bechet@irisa.fr

Annie Foret
Université de Rennes1, IRISA
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex
France
Annie.Foret@irisa.fr

Abstract

This paper is concerned with learning categorial grammars in Gold's model. In contrast to k-valued classical categorial grammars, k-valued Lambek grammars are not learnable from strings. This result was shown for several variants but the question was left open for the weakest one, the non-associative variant NL.

We show that the class of rigid and k-valued NL grammars is unlearnable from strings, for each k; this result is obtained by a specific construction of a limit point in the considered class, that does not use product operator.

Another interest of our construction is that it provides limit points for the whole hierarchy of Lambek grammars, including the recent pregroup grammars.

Such a result aims at clarifying the possible directions for future learning algorithms: it expresses the difficulty of learning categorial grammars from strings and the need for an adequate structure on examples.

1 Introduction

Categorial grammars (Bar-Hillel, 1953) and Lambek grammars (Lambek, 1958; Lambek, 1961) have been studied in the field of natural language processing. They are well adapted to learning perspectives since they are completely lexicalized and an actual way of research is to determine the sub-classes of such grammars that remain learnable in the sense of Gold (Gold, 1967).

We recall that learning here consists to define an algorithm on a finite set of sentences that converge to obtain a grammar in the class that generates the examples. Let $\mathcal{G}$ be a class of grammars, that we wish to learn from positive examples. Formally, let $\mathcal{L}(G)$ denote the language associated with grammar G, and let V be a given alphabet, a learning algorithm is a function ϕ from finite sets of words in V^* to $\mathcal{G}$, such that for all $G \in \mathcal{G}$ with $\mathcal{L}(G) =< e_i >_{i \in N}$ there exists a grammar $G' \in \mathcal{G}$ and there exists $n_0 \in N$ such that: $\forall n > n_0\ \phi(\{e_1, \ldots, e_n\}) = G' \in \mathcal{G}$ with $\mathcal{L}(G') = \mathcal{L}(G)$.

After pessimistic unlearnability results in (Gold, 1967), learnability of non trivial classes has been proved in (Angluin, 1980) and (Shinohara, 1990). Recent works from (Kanazawa, 1998) and (Nicolas, 1999) following (Buszkowski and Penn, 1990) have answered the problem for different sub-classes of classical categorial grammars (we recall that the whole class of classical categorial grammars is equivalent to context free grammars; the same holds for the class of Lambek grammars (Pentus, 1993) that is thus not learnable in Gold's model).

The extension of such results for Lambek grammars is an interesting challenge that is addressed by works on logic types from (Dudau-Sofronie et al., 2001) (these grammars enjoy a direct link with Montague semantics), learning from structures in (Retor and Bonato, september 2001), complexity results from (Florêncio, 2002) or unlearnability results from

(Foret and Le Nir, 2002a; Foret and Le Nir, 2002b); this result was shown for several variants but the question was left open for the basic variant, the non-associative variant *NL*.

In this paper, we consider the following question: is the non-associative variant *NL* of k-valued Lambek grammars learnable from strings; we answer by constructing a limit point for this class. Our construction is in some sense more complex than those for the other systems since they do not directly translate as limit point in the more restricted system *NL*.

The paper is organized as follows. Section 2 gives some background knowledge on three main aspects: Lambek categorial grammars ; learning in Gold's model ; Lambek pregroup grammars that we use later as models in some proofs. Section 3 then presents our main result on *NL* (*NL* denotes non-associative Lambek grammars not allowing empty sequence): after a construction overview, we discuss some corollaries and then provide the details of proof. Section 4 concludes.

2 Background

2.1 Categorial Grammars

The reader not familiar with Lambek Calculus and its non-associative version will find nice presentation in the first ones written by Lambek (Lambek, 1958; Lambek, 1961) or more recently in (Kandulski, 1988; Aarts and Trautwein, 1995; Buszkowski, 1997; Moortgat, 1997; de Groote, 1999; de Groote and Lamarche, 2002).

The *types* Tp, or formulas, are generated from a set of *primitive types* Pr, or atomic formulas by three binary connectives " / " (over), " \ " (under) and "•" (product): $Tp ::= Pr \mid Tp \setminus Tp \mid Tp \,/\, Tp \mid Tp \bullet Tp$. As a logical system, we use a Gentzen-style sequent presentation. A sequent $\Gamma \vdash A$ is composed of a sequence of formulas Γ which is the antecedent configuration and a succedent formula A.

Let Σ be a fixed alphabet. A *categorial grammar* over Σ is a finite relation G between Σ and Tp. If $< c, A > \in G$, we say that G *assigns* A to c, and we write $G : c \mapsto A$.

2.1.1 Lambek Derivation $\vdash_L$

The relation $\vdash_L$ is the smallest relation $\vdash$ between Tp^+ and Tp, such that for all $\Gamma, \Gamma' \in Tp^+, \Delta, \Delta' \in Tp^*$ and for all $A, B, C \in Tp$:

$$\frac{\Delta, A, \Delta' \vdash C \quad \Gamma \vdash A}{\Delta, \Gamma, \Delta' \vdash C}\,(Cut) \qquad A \vdash A \ (Id)$$

$$\frac{\Gamma \vdash A \quad \Delta, B, \Delta' \vdash C}{\Delta, B \,/\, A, \Gamma, \Delta' \vdash C}\,/L \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash B \,/\, A}\,/R$$

$$\frac{\Gamma \vdash A \quad \Delta, B, \Delta' \vdash C}{\Delta, \Gamma, A \setminus B, \Delta' \vdash C}\,\setminus L \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \setminus B}\,\setminus R$$

$$\frac{\Delta, A, B, \Delta' \vdash C}{\Delta, A \bullet B, \Delta' \vdash C}\,\bullet L \qquad \frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \bullet B}\,\bullet R$$

We write $L_\emptyset$ for the Lambek calculus with empty antecedents (left part of the sequent).

2.1.2 Non-associative Lambek Derivation $\vdash_{NL}$

In the Gentzen presentation, the derivability relation of *NL* holds between a term in $\mathcal{S}$ and a formula in Tp, where the term language is $\mathcal{S} ::= Tp|(\mathcal{S}, \mathcal{S})$. Terms in $\mathcal{S}$ are also called *G-terms*. A sequent is a pair $(\Gamma, A) \in \mathcal{S} \times Tp$. The notation $\Gamma[\Delta]$ represents a G-term with a distinguished occurrence of Δ (with the same position in premise and conclusion of a rule). The relation $\vdash_{NL}$ is the smallest relation $\vdash$ between $\mathcal{S}$ and Tp, such that for all $\Gamma, \Delta \in \mathcal{S}$ and for all $A, B, C \in Tp$:

$$\frac{\Gamma[A] \vdash C \quad \Delta \vdash A}{\Gamma[\Delta] \vdash C}\,(Cut) \qquad A \vdash A \ (Id)$$

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

$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(\Gamma, A \setminus B)] \vdash C}\,\setminus L \qquad \frac{(A, \Gamma) \vdash B}{\Gamma \vdash A \setminus B}\,\setminus R$$

$$\frac{\Delta[(A, B)] \vdash C}{\Delta[A \bullet B] \vdash C}\,\bullet L \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma, \Delta) \vdash (A \bullet B)}\,\bullet R$$

We write $NL_\emptyset$ for the non-associative Lambek calculus with empty antecedents (left part of the sequent).

2.1.3 Notes

Cut elimination. We recall that cut rule can be eliminated in $\vdash_L$ and $\vdash_{NL}$: every derivable sequent has a cut-free derivation.

Type order. The order $ord(A)$ of a type A of L or NL is defined by:

$ord(A) = 0$ if A is a primitive type
$ord(C_1 \,/\, C_2) = max(ord(C_1), ord(C_2) + 1)$
$ord(C_1 \setminus C_2) = max(ord(C_1) + 1, ord(C_2))$
$ord(C_1 \bullet C_2) = max(ord(C_1), ord(C_2))$

2.1.4 Language.

Let G be a categorial grammar over Σ. G *generates* a string $c_1 \ldots c_n \in \Sigma^+$ iff there are types $A_1, \ldots, A_n \in Tp$ such that: $G : c_i \mapsto A_i$ $(1 \leq i \leq n)$ and $A_1, \ldots, A_n \vdash_L S$. The *language of* G, written $\mathcal{L}_L(G)$ is the set of strings generated by G. We define similarly $\mathcal{L}_{L_\emptyset}(G)$, $\mathcal{L}_{NL}(G)$ and $\mathcal{L}_{NL_\emptyset}(G)$ replacing $\vdash_L$ by $\vdash_{L_\emptyset}$, $\vdash_{NL}$ and $\vdash_{NL_\emptyset}$ in the sequent where the types are parenthesized in some way.

2.1.5 Notation.

In some sections, we may write simply $\vdash$ instead of $\vdash_L, \vdash_{L_\emptyset}, \vdash_{NL}$ or $\vdash_{NL_\emptyset}$. We may simply write $\mathcal{L}(G)$ accordingly.

2.1.6 Rigid and k-valued Grammars.

Categorial grammars that assign at most k types to each symbol in the alphabet are called *k-valued grammars*; 1-valued grammars are also called *rigid* grammars.

Example 1 *Let* $\Sigma_1 = \{John, Mary, likes\}$ *and let* $Pr = \{S, N\}$ *for sentences and nouns respectively. Let* $G_1 = \{John \mapsto N,\ Mary \mapsto N,\ likes \mapsto N \setminus (S \,/\, N)\}$. *We get* $(John\ likes\ Mary) \in \mathcal{L}_{NL}(G_1)$ *since* $((N,\ N \setminus (S \,/\, N)),\ N) \vdash_{NL} S$. G_1 *is a rigid (or 1-valued) grammar.*

2.2 Learning and Limit Points

We now recall some useful definitions and known properties on learning.

2.2.1 Limit Points

A class $\mathcal{CL}$ of languages has *a limit point* iff there exists an infinite sequence $< L_n >_{n \in N}$ of languages in $\mathcal{CL}$ and a language $L \in \mathcal{CL}$ such that: $L_0 \subsetneq L_1 \ldots \subsetneq \ldots \subsetneq L_n \subsetneq \ldots$ and $L = \bigcup_{n \in N} L_n$ (L is *a limit point* of $\mathcal{CL}$).

2.2.2 Limit Points Imply Unlearnability

The following property is important for our purpose. If the languages of the grammars in a class $\mathcal{G}$ have a limit point then the class $\mathcal{G}$ is *unlearnable*. [1]

2.3 Some Useful Models

For ease of proof, in next section we use two kinds of models that we now recall: free groups and pregroups introduced recently by (Lambek, 1999) as an alternative of existing type grammars.

2.3.1 Free Group Interpretation.

Let FG denote the free group with generators Pr, operation $\cdot$ and with neutral element 1. We associate with each formula C of L or NL, an element in FG written $[\![C]\!]$ as follows:

$[\![A]\!] = A$ if A is a primitive type
$[\![C_1 \setminus C_2]\!] = [\![C_1]\!]^{-1} \cdot [\![C_2]\!]$
$[\![C_1 \,/\, C_2]\!] = [\![C_1]\!] \cdot [\![C_2]\!]^{-1}$
$[\![C_1 \bullet C_2]\!] = [\![C_1]\!] \cdot [\![C_2]\!]$

We extend the notation to sequents by:

$[\![C_1, C_2, \ldots, C_n]\!] = [\![C_1]\!] \cdot [\![C_2]\!] \cdot \ \cdots \ \cdot [\![C_n]\!]$

The following property states that FG is a model for L (hence for NL): if $\Gamma \vdash_L C$ then $[\![\Gamma]\!] =_{FG} [\![C]\!]$

2.3.2 Free Pregroup Interpretation

Pregroup. A *pregroup* is a structure $(P, \leq, \cdot, l, r, 1)$ such that $(P, \leq, \cdot, 1)$ is a partially ordered monoid[2] and l, r are two unary operations on P that satisfy for all $a \in P$ $a^l a \leq 1 \leq a a^l$ and $a a^r \leq 1 \leq a^r a$.

Free pregroup. Let $(P, \leq)$ be an ordered set of primitive types, $P^{(\mathbb{Z})} = \{p^{(i)} \mid p \in P, i \in \mathbb{Z}\}$ is the set of atomic types and $T_{(P,\leq)} = \left(P^{(\mathbb{Z})}\right)^* = \{p_1^{(i_1)} \cdots p_n^{(i_n)} \mid 0 \leq k \leq n, p_k \in P \text{ and } i_k \in \mathbb{Z}\}$ is the set of types. For X and $Y \in T_{(P,\leq)}$, $X \leq Y$ iif this relation is deductible in the following system where $p, q \in P$, $n, k \in \mathbb{Z}$ and $X, Y, Z \in T_{(P,\leq)}$:

---

[1] This implies that the class has infinite elasticity. A class $\mathcal{CL}$ of languages has *infinite elasticity* iff $\exists < e_i >_{i \in N}$ sentences $\exists < L_i >_{i \in N}$ languages in $\mathcal{CL}$ $\forall i \in N$: $e_i \notin L_i$ and $\{e_1, \ldots, e_n\} \subseteq L_{n+1}$.

[2] We briefly recall that a *monoid* is a structure $< M, \cdot, 1 >$, such that $\cdot$ is associative and has a neutral element 1 ($\forall x \in M : 1 \cdot x = x \cdot 1 = x$). A partially ordered monoid is a monoid $M, \cdot, 1)$ with a partial order $\leq$ that satisfies $\forall a, b, c$: $a \leq b \Rightarrow c \cdot a \leq c \cdot b$ and $a \cdot c \leq b \cdot c$.

$$X \leq X \ (Id) \qquad \frac{X \leq Y \quad Y \leq Z}{X \leq Z}\ (Cut)$$

$$\frac{XY \leq Z}{Xp^{(n)}p^{(n+1)}Y \leq Z}\ (A_L) \qquad \frac{X \leq YZ}{X \leq Yp^{(n+1)}p^{(n)}Z}\ (A_R)$$

$$\frac{Xp^{(k)}Y \leq Z}{Xq^{(k)}Y \leq Z}\ (IND_L) \qquad \frac{X \leq Yp^{(k)}Z}{X \leq Yq^{(k)}Z}\ (IND_R)$$

$q \leq p$ if k is even, and $p \leq q$ if k is odd

This construction, proposed by Buskowski, defines a pregroup that extends $\leq$ on primitive types P to $T_{(P,\leq)}$[3].

Cut elimination. As for L and NL, cut rule can be eliminated: every derivable inequality has a cut-free derivation.

Simple free pregroup. A *simple free pregroup* is a free pregroup where the order on primitive type is equality.

Free pregroup interpretation. Let *FP* denotes the simple free pregroup with Pr as primitive types. We associate with each formula C of L or NL, an element in *FP* written $[C]$ as follows:

$[A] = A$ if A is a primitive type
$[C_1 \setminus C_2] = [C_1]^r[C_2]$
$[C_1 \ / \ C_2] = [C_1][C_2]^l$
$[C_1 \bullet C_2] = [C_1][C_2]$

We extend the notation to sequents by:
$[A_1, \dots, A_n] = [A_1] \cdots [A_n]$

The following property states that *FP* is a model for L (hence for NL): if $\Gamma \vdash_L C$ then $[\Gamma] \leq_{FP} [C]$.

3 Limit Point Construction

3.1 Method overview and remarks

Form of grammars. We define grammars G_n where A, B, D_n and E_n are complex types and S is the main type of each grammar:

$G_n = \{a \mapsto A \ / \ B; \ \ b \mapsto D_n; \ \ c \mapsto E_n \setminus S\}$

Some key points.

- We prove that $\{a^k bc \mid 0 \leq k \leq n\} \subseteq \mathcal{L}(G_n)$ using the following properties:

  $B \vdash A$ (but $A \not\vdash B$)
  $(A \ / \ B, D_{n+1}) \vdash D_n$
  $D_n \vdash E_n$
  $E_n \vdash E_{n+1}$

  we get:

  $bc \in \mathcal{L}(G_n)$ since $D_n \vdash E_n$
  if $w \in \mathcal{L}(G_n)$ then $aw \in \mathcal{L}(G_{n+1})$ since $(A \ / \ B, D_{n+1}) \vdash D_n \vdash E_n \vdash E_{n+1}$

- The condition $A \not\vdash B$ is crucial for strictness of language inclusion. In particular: $(A \ / \ B, A) \not\vdash A$, where $A = D_0$

- This construction is in some sense more complex than those for the other systems (Foret and Le Nir, 2002a; Foret and Le Nir, 2002b) since they do not directly translate as limit points in the more restricted system NL.

[3]Left and right adjoints are defined by $(p^{(n)})^l = p^{(n-1)}$, $(p^{(n)})^r = p^{(n+1)}$, $(XY)^l = Y^l X^l$ and $(XY)^r = Y^r X^r$. We write p for $p^{(0)}$.

3.2 Definition and Main Results

Definitions of Rigid grammars G_n and G_*

Definition 1 *Let p, q, S, three primitive types. We define:*

$A = D_0 = E_0 = q \ / \ (p \setminus q)$
$B = p$
$D_{n+1} = (A \ / \ B) \setminus D_n$
$E_{n+1} = (A \ / \ A) \setminus E_n$

Let $G_n = \left\{ \begin{array}{l} a \mapsto A \ / \ B = (q \ / \ (p \setminus q)) \ / \ p \\ b \mapsto D_n \\ c \mapsto E_n \setminus S \end{array} \right\}$

Let $G_* = \{a \mapsto (p \ / \ p) \quad b \mapsto p \quad c \mapsto (p \setminus S)\}$

Main Properties

Proposition 1 (language description)

- $\mathcal{L}(G_n) = \{a^k bc \mid 0 \leq k \leq n\}$
- $\mathcal{L}(G_*) = \{a^k bc \mid 0 \leq k\}$.

From this construction we get a limit point and the following result.

Proposition 2 (NL-non-learnability) *The class of languages of rigid (or k-valued for an arbitrary k) non-associative Lambek grammars (not allowing empty sequence and without product) admits a limit point ; the class of rigid (or k-valued for an arbitrary k) non-associative Lambek grammars (not allowing empty sequence and without product) is not learnable from strings.*

3.3 Details of proof for G_n

Lemma

$\{a^k bc \mid 0 \le k \le n\} \subseteq \mathcal{L}(G_n)$

Proof: It is relatively easy to see that for $0 \le k \le n$, $a^k bc \in \mathcal{L}(G_n)$. We have to consider $((\underbrace{a \cdots (a(a}_{k}\, b)) \cdots)c)$ and prove the following sequent in *NL*:

$$(\underbrace{\overbrace{(((A / B), \ldots , ((A / B),}^{(a\cdots(a}}_{k}$$

$$\overbrace{\underbrace{((A / B) \setminus \cdots \setminus ((A / B) \setminus}_{n} A) \cdots)}^{b}, \cdots),$$

$$\overbrace{\underbrace{((A / A) \setminus \cdots \setminus ((A / A) \setminus}_{n} A) \cdots) \setminus S)}^{c}) \vdash_{NL} S$$

Models of *NL*

For the converse, (for technical reasons and to ease proofs) we use both free group and free pregroup models of *NL* since a sequent is valid in *NL* only if its interpretation is valid in both models.

Translation in free groups

The free group translation for the types of G_n is:

$[\![p]\!] = p$, $[\![q]\!] = q$, $[\![S]\!] = S$
$[\![x / y]\!] = [\![x]\!] \cdot [\![y]\!]^{-1}$
$[\![x \setminus y]\!] = [\![x]\!]^{-1} \cdot [\![y]\!]$
$[\![x \bullet y]\!] = [\![x]\!] \cdot [\![y]\!]$

Type-raising disappears by translation:

$[\![x / (y \setminus x)]\!] = [\![x]\!] \cdot ([\![y]\!]^{-1} \cdot [\![x]\!])^{-1} = [\![y]\!]$

Thus, we get :

$[\![A]\!] = [\![D_0]\!] = [\![E_0]\!] = [\![q / (p \setminus q)]\!] = p$
$[\![B]\!] = p$
$[\![A / B]\!] = [\![A]\!] \cdot [\![B]\!]^{-1} = pp^{-1} = 1$
$[\![D_{n+1}]\!] = [\![(A / B) \setminus D_n]\!] = [\![D_n]\!] = [\![D_0]\!] = p$
$[\![E_{n+1}]\!] = [\![(A / A) \setminus E_n]\!] = [\![E_n]\!] = [\![E_0]\!] = p$

Translation in free pregroups

The free pregroup translation for the types of G_n is:

$[p] = p$, $[q] = q$, $[S] = S$
$[x \setminus y] = [x]^r[y]$
$[y / x] = [y][x]^l$
$[x \bullet y] = [x][y]$

Type-raising translation:

$[x / (y \setminus x)] = [x]([y]^r[x])^l = [x][x]^l[y]$
$[x / (x \setminus x)] = [x]([x]^r[x])^l = [x][x]^l[x] = [x]$

Thus, we get:

$[A] = [D_0] = [E_0] = [q / (p \setminus q)] = qq^lp$
$[B] = p$
$[A / B] = [A][B]^l = qq^lpp^l$
$[D_{n+1}] = [(A / B)]^r[D_n] = \underbrace{pp^rqq^r}_{n+1} qq^lp$
$[E_{n+1}] = [(A / A) \setminus E_n] = [A][A]^lqq^lp = qq^lp$

Lemma

$\mathcal{L}(G_n) \subseteq \{a^kba^{k'}ca^{k''}; 0 \le k, 0 \le k', 0 \le k''\}$

Proof: Let τ_n denote the type assignment by the rigid grammar G_n. Suppose $\tau_n(w) \vdash S$, using free groups $[\![\tau_n(w)]\!] = S$;
- This entails that w has exactly one occurrence of c (since $[\![\tau_n(c)]\!] = p^{-1}S$ and the other type images are either 1 or p)
- Then, this entails that w has exactly one occurrence of b on the left of the occurrence of c (since $[\![\tau_n(c)]\!] = p^{-1}S$, $[\![\tau_n(b)]\!] = p$ and $[\![\tau_n(a)]\!] = 1$)

Lemma

$\mathcal{L}(G_n) \subseteq \{a^kbc \mid 0 \le k\}$

Proof: Suppose $\tau_n(w) \vdash S$, using pregroups $[\tau_n(w)] \le S$. We can write $w = a^kba^{k'}ca^{k''}$ for some k, k', k'', such that:

$$[\tau_n(w)] = \underbrace{qq^lpp^l}_{k}\underbrace{pp^rqq^r}_{n} qq^lp \underbrace{qq^lpp^l}_{k'} p^rqq^rS \underbrace{qq^lpp^l}_{k''}$$

For $q = 1$, we get $\underbrace{pp^l}_{k}\underbrace{pp^r}_{n} p \underbrace{pp^l}_{k'} p^rS \underbrace{pp^l}_{k''} \le S$ and it yields $p \underbrace{pp^l}_{k'} p^rS \underbrace{pp^l}_{k''} \le S$.

We now discuss possible deductions (note that $pp^lpp^l \cdots pp^l = pp^l$):

- if k' and $k'' \neq 0$: $ppp^lp^rSpp^l \le S$ impossible.
- if $k' \neq 0$ and $k'' = 0$: $ppp^lp^rS \le S$ impossible.
- if $k' = 0$ and $k'' \neq 0$: $pp^rSpp^l \le S$ impossible.
- if $k' = k'' = 0$: $w \in \{a^kbc \mid 0 \le k\}$ ■

(Final) Lemma

$\mathcal{L}(G_n) \subseteq \{a^kbc \mid 0 \le k \le n\}$

Proof: Suppose $\tau_n(w) \vdash S$, using pregroups $[\tau_n(w)] \leq S$. We can write $w = a^k bc$ for some k, such that :

$$[\tau_n(w)] = \underbrace{qq^l pp^l}_{k} \underbrace{pp^r qq^r}_{n} qq^l pp^r qq^r S$$

We use the following property (its proof is in Appendix A) that entails that $0 \leq k \leq n$.

(Auxiliary) Lemma:

if (1) $X, Y, qq^l p, p^r qq^r, S \leq S$

where $X \in \{pp^l, qq^l\}^*$ and $Y \in \{qq^r, pp^r\}^*$

then $\begin{cases} (2) & nbalt(Xqq^l) \leq nbalt(qq^r Y) \\ (2bis) & nbalt(Xpp^l) \leq nbalt(pp^r Y) \end{cases}$

where $nbalt$ counts the alternations of p's and q's sequences *(forgetting/dropping their exponents).*

3.4 Details of proof for G_*

Lemma

$\{a^k bc \mid 0 \leq k\} \subseteq \mathcal{L}(G_*)$

Proof: As with G_n, it is relatively easy to see that for $k \geq 0$, $a^k bc \in \mathcal{L}(G_*)$. We have to consider $((\underbrace{a \cdots (a(a}_{k} b)) \cdots)c)$ and prove the following sequent in NL:

$$(\underbrace{(((p \,/\, p), \ldots , ((p \,/\, p),}_{k} p) \cdots), (p \setminus S)) \vdash_{NL} S$$

Lemma

$\mathcal{L}(G_*) \subseteq \{a^k bc \mid 0 \leq k\}$

Proof: Like for $w \in G_n$, due to free groups, a word of $\mathcal{L}(G_*)$ has exactly one occurrence of c and one occurrence of b on the left of c (since $[\![\tau_*(c)]\!] = p^{-1}S$, $[\![\tau_*(b)]\!] = p$ and $[\![\tau_*(a)]\!] = 1$).

Suppose $w = a^k ba^{k'} ca^{k''}$ a similar discussion as for G_n in pregroups, gives $k' = k'' = 0$, hence the result ■

3.5 Non-learnability of a Hierarchy of Systems

An interest point of this construction: It provides a limit point for the whole hierarchy of Lambek grammars, and pregroup grammars.

Limit point for pregroups

The translation $[\cdot]$ of G_n gives a limit point for the simple free pregroup since for $i \in \{*, 0, 1, 2, \ldots\}$:
$\tau_i(w) \vdash_{NL} S$ iff $w \in \mathcal{L}_{NL}(G_i)$ by definition ;
$\tau_i(w) \vdash_{NL} S$ implies $[\tau_i(w)] \leq S$ by models ;
$[\tau_i(w)] \leq S$ implies $w \in \mathcal{L}_{NL}(G_i)$ from above.

Limit point for $NL_\emptyset$

The same grammars and languages work since for $i \in \{*, 0, 1, 2, \ldots\}$:
$\tau_i(w) \vdash_{NL} S$ iff $[\tau_i(w)] \leq S$ from above ;
$\tau_i(w) \vdash_{NL} S$ implies $\tau_i(w) \vdash_{NL_\emptyset} S$ by hierarchy ;
$\tau_i(w) \vdash_{NL_\emptyset} S$ implies $[\tau_i(w)] \leq S$ by models.

Limit point for L and $L_\emptyset$

The same grammars and languages work since for $i \in \{*, 0, 1, 2, \ldots\}$:
$\tau_i(w) \vdash_{NL} S$ iff $[\tau_i(w)] \leq S$ from above ;
$\tau_i(w) \vdash_{NL} S$ implies $\tau_i(w) \vdash_{L} S$ using hierarchy ;
$\tau_i(w) \vdash_{L} S$ implies $\tau_i(w) \vdash_{L_\emptyset} S$ using hierarchy ;
$\tau_i(w) \vdash_{L_\emptyset} S$ implies $[\tau_i(w)] \leq S$ by models.

To summarize : $w \in \mathcal{L}_{NL}(G_i)$ iff $[\tau_i(w)] \leq S$ iff $w \in \mathcal{L}_{NL_\emptyset}(G_i)$ iff $w \in \mathcal{L}_{L}(G_i)$ iff $w \in \mathcal{L}_{L\emptyset}(G_i)$

4 Conclusion and Remarks

Lambek grammars. We have shown that without empty sequence, non-associative Lambek rigid grammars are not learnable from strings. With this result, the whole landscape of Lambek-like rigid grammars (or k-valued for an arbitrary k) is now described as for the learnability question (from strings, in Gold's model).

Non-learnability for subclasses. Our construct is of order 5 and does not use the product operator. Thus, we have the following corollaries:

- Restricted connectives: k-valued NL, $NL_\emptyset$, L and $L_\emptyset$ grammars **without product** are not learnable from strings.
- Restricted type order:
  - k-valued NL, $NL_\emptyset$, L and $L_\emptyset$ grammars (without product) with types **not greater than order 5** are not learnable from strings[4].
  - k-valued free pregroup grammars with types **not greater than order 1** are not learnable from strings[5].

The learnability question may still be raised for NL grammars of order lower than 5.

[4]Even less for some systems. For example in $L_\emptyset$, all E_n collapse to A

[5]The order of a type $p_1^{i_1} \cdots p_k^{i_k}$ is the maximum of the absolute value of the exponents: $max(|i_1|, \ldots , |i_k|)$.

Special learnable subclasses. Note that however, we get specific learnable subclasses of k-valued grammars when we consider NL, $NL_\emptyset$, L or $L_\emptyset$ without product and we bind the order of types in grammars to be not greater than 1. This holds for all variants of Lambek grammars as a corollary of the equivalence between generation in classical categorial grammars and in Lambek systems for grammars with such product-free types (Buszkowski, 2001).

Restriction on types. An interesting perspective for learnability results might be to introduce reasonable restrictions on types. From what we have seen, the order of type alone (order 1 excepted) does not seem to be an appropriate measure in that context.

Structured examples. These results also indicate the necessity of using structured examples as input of learning algorithms. What intermediate structure should then be taken as a good alternative between insufficient structures (strings) and linguistic unrealistic structures (full proof tree structures) remains an interesting challenge.

References

E. Aarts and K. Trautwein. 1995. Non-associative Lambek categorial grammar in polynomial time. *Mathematical Logic Quaterly*, 41:476–484.

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

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

Wojciech Buszkowski and Gerald Penn. 1990. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454.

W. Buszkowski. 1997. Mathematical linguistics and proof theory. In van Benthem and ter Meulen (van Benthem and ter Meulen, 1997), chapter 12, pages 683–736.

Wojciech Buszkowski. 2001. Lambek grammars based on pregroups. In Philippe de Groote, Glyn Morill, and Christian Retoré, editors, *Logical aspects of computational linguistics: 4th International Conference, LACL 2001, Le Croisic, France, June 2001*, volume 2099. Springer-Verlag.

Philippe de Groote and François Lamarche. 2002. Classical non-associative lambek calculus. *Studia Logica*, 71.1 (2).

Philippe de Groote. 1999. Non-associative Lambek calculus in polynomial time. In *$8^t h$ Workshop on theorem proving with analytic tableaux and related methods*, number 1617 in Lecture Notes in Artificial Intelligence. Springer-Verlag, March.

Dudau-Sofronie, Tellier, and Tommasi. 2001. Learning categorial grammars from semantic types. In *13th Amsterdam Colloquium*.

C. Costa Florêncio. 2002. Consistent Identification in the Limit of the Class k-valued is NP-hard. In *LACL*.

Annie Foret and Yannick Le Nir. 2002a. Lambek rigid grammars are not learnable from strings. In *COLING'2002, 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Annie Foret and Yannick Le Nir. 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.

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

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

Maciej Kandulski. 1988. The non-associative lambek calculus. In W. Marciszewski W. Buszkowski and J. Van Bentem, editors, *Categorial Grammar*, pages 141–152. Benjamins, Amsterdam.

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

Joachim Lambek. 1961. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society.

J. Lambek. 1999. Type grammars revisited. In Alain Lecomte, François Lamarche, and Guy Perrier, editors, *Logical aspects of computational linguistics: Second International Conference, LACL '97, Nancy, France, September 22–24, 1997; selected papers*, volume 1582. Springer-Verlag.

Michael Moortgat. 1997. Categorial type logic. In van Benthem and ter Meulen (van Benthem and ter Meulen, 1997), chapter 2, pages 93–177.

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

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

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

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

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

Appendix A. Proof of Auxiliary Lemma

(Auxiliary) Lemma:

if (1) $XYqq^lpp^rqq^rS \leq S$

where $X \in \{pp^l, qq^l\}^*$ and $Y \in \{qq^r, pp^r\}^*$

$$\text{then } \begin{cases} (2) & nbalt(Xqq^l) \leq nbalt(qq^rY) \\ (2bis) & nbalt(Xpp^l) \leq nbalt(pp^rY) \end{cases}$$

where $nbalt$ counts the alternations of p's and q's sequences *(forgetting/dropping their exponents)*.

Proof: By induction on derivations in Gentzen style presentation of free pregroups (without $\mathbf{Cut}$). Suppose $XYZS \leq S$

$$\text{where } \begin{cases} X \in \{pp^l, qq^l\}^* \\ Y \in \{qq^r, pp^r\}^* \\ Z \in \{(qq^lpp^rqq^r), (qq^lqq^r), (qq^r), 1\} \end{cases}$$

$$\text{We show that } \begin{cases} nbalt(Xqq^l) \leq nbalt(qq^rY) \\ nbalt(Xpp^l) \leq nbalt(pp^rY) \end{cases}$$

The last inference rule can only be (A_L)

- Case (A_L) on X: The antecedent is similar with X' instead of X, where X is obtained from X' by insertion (in fact inserting q^lq in the middle of qq^l as the replacement of qq^l with qq^lqq^l or similarly with p instead of q).
  - By such an insertion: (i) $nbalt(X'qq^l) = nbalt(Xqq^l)$ (similar for p).
  - By induction hypothesis: (ii) $nbalt(X'qq^l) \leq nbalt(qq^rY)$ (similar for p).
  - Therefore from (i) (ii): $nbalt(Xqq^l) \leq nbalt(qq^rY)$ (similar for p).
- Case (A_L) on Y: The antecedent is $XY'ZS \leq S$ where Y is obtained from Y' by insertion (in fact insertion of pp^r or qq^r), such that $Y' \in \{pp^r, qq^r\}^*$. Therefore the induction applies $nbalt(Xqq^l) \leq nbalt(qq^rY')$ and $nbalt(qq^rY) \geq nbalt(qq^rY')$ (similar for p) hence the result.
- Case (A_L) on Z (Z non empty):
  - if $Z = (qq^lpp^rqq^r)$ the antecedent is $XYZ'S \leq S$, where $Z' = qq^lqq^r$.
  - if $Z = (qq^lqq^r)$ the antecedent is $XYZ'S \leq S$, where $Z' = qq^r$;
  - if $Z = (qq^r)$ the antecedent is $XYZ'S \leq S$, where $Z' = \epsilon$.

  In all three cases the hypothesis applies to XYZ' and gives the relationship between X and Y.
- case (A_L) between X and Y: Either $X = X''qq^l$ and $Y = qq^rY''$ or $X = X''pp^l$ and $Y = pp^rY''$. In the q case, the last inference step is the introduction of q^lq:

$$\frac{X''qq^rY''ZS \leq S}{\underbrace{X''qq^l}_{X}\underbrace{qq^rY''}_{Y}ZS \leq S}$$

We now detail the q case. The antecedent can be rewritten as $X''YZS \leq S$ and we have: (i)

$$\begin{array}{rcl} nbalt(Xqq^l) & = & nbalt(X''qq^lqq^l) \\ & = & nbalt(X''qq^l) \\ nbalt(Xpp^l) & = & nbalt(X''qq^lpp^l) \\ & = & 1 + nbalt(X''qq^l) \\ nbalt(qq^rY) & = & nbalt(qq^rqq^rY'') \\ & = & nbalt(qq^rY'') \\ nbalt(pp^rY) & = & nbalt(pp^rqq^rY'') \\ & = & 1 + nbalt(qq^rY'') \end{array}$$

We can apply the induction hypothesis to $X''YZS \leq S$ and get (ii):

$$nbalt(X''qq^l) \leq nbalt(qq^rY)$$

Finally from (i) (ii) and the induction hypothesis:

$$\begin{array}{rcl} \underline{nbalt(Xqq^l)} & = & nbalt(X''qq^l) \\ & \leq & \underline{nbalt(qq^rY)} \\ \underline{nbalt(Xpp^l)} & = & 1 + nbalt(X''qq^l) \\ & \leq & 1 + nbalt(qq^rY) \\ & = & 1 + nbalt(qq^rqq^rY'') \\ & = & 1 + nbalt(qq^rY'') \\ & = & \underline{nbalt(pp^rY)} \end{array}$$

The second case with p instead of q is similar.